平成 14 年度　東京大学大学院工学系研究科　修士論文

実世界における移動ロボットのための環境の変化に強い自己位置推定と行動決定

Self-localization and Decision Making Method
Robust Against Environment Change
for Real World Mobile Robots

指導教官　　太田 順　助教授

東京大学大学院　工学系研究科　精密機械工学専攻
学生証番号　16237

上田 隆一

目 次

第1章　序論

1.1　ロボカップ

ロボカップ（Robot Soccer World Cup，RoboCup）は，実世界における人工知能とロボット研究のための標準問題として，日本の研究者たちによって提案された，ロボットサッカーの世界大会である [浅田 00, 松原 02].

かつて，人工知能の標準問題としてコンピュータチェスがあった．コンピュータチェスでは,「人間のグランドマスターを打ち負かすコンピュータプログラムの作成」が最終目標であり，1997 年にその目標は達成された．そして，コンピュータチェスのプログラムを作成する過程で開発されたアルゴリズムやアーキテクチャは，幅広い分野で現在利用されており，この事実は，目標が達成されたことよりも重要と言える．

ロボカップにも同様の側面がある．すなわち，最終目標を,「FIFA ルールに従い，2050 年までに自律型のヒューマノイドロボット 11 台で人間のワールドカップチャンピオンを打破する」ことと定め，そのための個々の問題に対して多数の研究者[*1]が一斉に取り組むことで，問題解決のための優れた手法を生み出そうというのである．これらの手法は，ロボットサッカーだけではなく，人工知能やロボット工学の分野においても有益であることが期待できる．

[浅田 97] では，標準問題としてサッカーが選ばれている理由として，以下の点が挙げられている．

(1) 世界で最も有名な競技であり，ルールが理解されている．

(2) 設計されたサッカーエージェントの優劣が，獲得点数という極めて明解な評価基準で判断できる．

(3) 最適解の発見は困難だが，現状の技術である程度実現可能である．

(4) チェスなどの，これまで人工知能の分野で扱われてきた標準問題にはない，**実世界，実時間性，協調の必要性**という，新たな問題を扱える．

4 で述べた問題は，将来ロボットが家庭やオフィスなどの日常世界で活動するために不可欠な要素技術であり，ロボカップの問題設定は，標準問題としての重要な性質を備えていると言える．

[*1] 現在，世界 35ヶ国，3000 人以上もの研究者，学生が参加している．

ロボカップは，使用するロボットや環境等が異なる，以下のような複数のリーグから構成されている．リーグで使用する環境やロボット，ルールによって，研究対象とする課題が異なる．

小型ロボットリーグ 卓球台の大きさのフィールドで，直径 180[mm] 以内の大きさのロボット 5 台までで戦う．このサイズでは，センサや CPU をオンボードにするのは困難なので，天井カメラの使用を許し，リモートブレイン方式を採用している．これは，知的交通システム（ITS）などで，複数のカメラを用いて乗り物を制御する課題に対応している．

中型ロボットリーグ 完全自律ロボットが，卓球台 9 台分（3×3）の大きさの競技場で戦う．1 チームのロボットは 5 台で，実時間画像処理，行動学習，協調行動の獲得などが課題となっている．

4 足ロボットリーグ 著者が参加しているリーグである．当リーグにおいて研究課題となることについては，次節で詳しく説明する．

ヒューマノイドリーグ 2002 年度福岡大会から追加されたリーグである．まだサッカーの試合ができるほどのレベルではなく，2002 年度は PK 合戦やダンスなどが行われた．

シミュレーションリーグ 計算機上に仮想的なフィールドを用意し，参加者は作成したプログラムをサーバに接続して，フィールド上のプレーヤーを制御する．このリーグでは，実機では手間のかかる研究や，多数のエージェントの協調に関する研究が扱われる．

このようにロボカップは現在，様々なリーグに分かれているが，それぞれのリーグで，研究者は共通の問題設定でハードウェアやソフトウェアを開発し，評価を行っている．また，サッカーの他に，大規模災害へのロボットの応用としてロボカップ・レスキュー，次世代の技術の担い手を育てるロボカップ・ジュニアなどの活動が行われている．

1.2　4 足ロボットリーグ

筆者らは，ロボカップの一部門ある 4 足ロボットリーグ [大橋 02] を題材として研究を行っている．このリーグでは，Fig.1.1 に見られる脚型ロボット ERS-210（SONY 製）を共通のハードウェアとして使用している．当リーグでは，ハードウェアの改造が禁止されており，純粋にソフトウェアの性能が競われている．

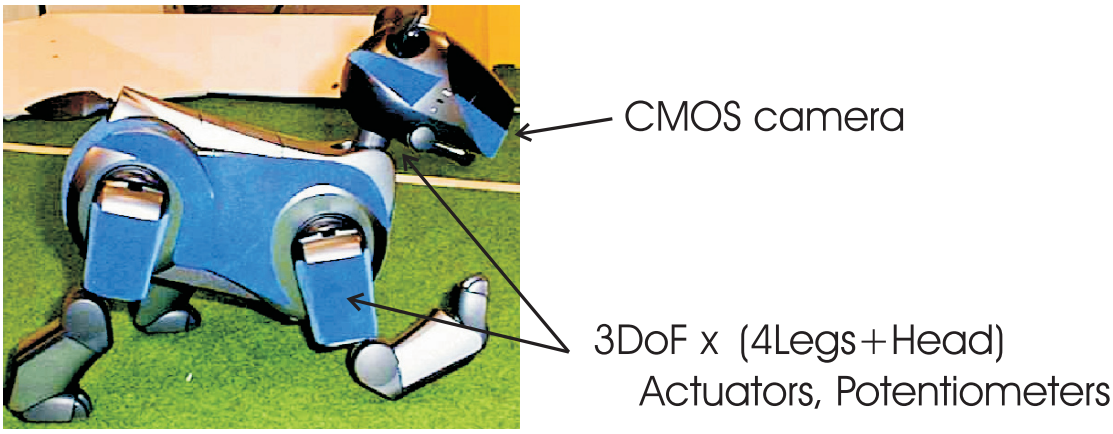

Fig. 1.1　ERS-210

ERS-210 は，以下のような特徴をもつロボットである．

多自由度　頭部に 3 自由度，各脚に 3 自由度ずつ，耳，尾にも自由度を持っている．各関節の制御能力はそれほど高くなく，足の形状が複雑なため，設計された歩行の移動距離，方向の再現性が低いことが問題とされていた．しかし，現在は再現性の高い歩行や全方向移動の歩行などの優れた歩行法が開発され，解決されつつある [Hengst 01]．

センシング　主たる外界センサは，鼻先に搭載されたカラー CMOS カメラである．カメラの視野は 58×48[deg] と，人間に比べると狭い．また，カメラ画像はロボットのメモリに 32[msec] 毎に送られるが，このときの画素数は 176×144[pixel] であり，低解像度の画像である．

計算資源　搭載されている CPU は MIPS192MHz，RAM は 32MB である．また，プログラムを PC からロボットへ移すための不揮発性メモリーの容量は 16MB である．

4 足ロボットリーグでは，現在のところセンシング能力の不足を克服し，情報処理のロバスト性を高めることで，ERS-210 を優れたサッカーロボットとする研究が多く行われている．ERS-210 に搭載されたカメラの視野は，人間の視野と比較すると非常に狭く，フィールド上の複数の物体を同時にカメラで捉える頻度は少ないため，ロボットは常に部分的な情報から自身のとるべき行動を決定しなければならない．この問題の克服のためには，得られた時間の異なる複数の画像から，有効な情報を抽出するというアプローチや，部分的な視覚情報から，妥当な行動を選択するアプローチなどが考えられる．また，あるフィールドで開発された画像処理アルゴリズムが，照明条件の異なる他のフィールドでも機能することは，ロボットが安定した性能を発揮するために必要となる．他リーグの最近の例を挙げると，ボールの色[*2]を抽出する方法ではなく，丸いという情報のみからボールを検出するという研究が行われている [Hanek 02]．

上記のような研究がロボカップで実用されるようになると，ロボカップの環境やルールは，実世界に近づくように改正される．これは，情報を得ることや行動することが，より困難になることを意味する．研究者は，それに対する対処法を研究し，克服することによって，ロボットを実世界で知的に行動させることに貢献できる．このように，ロボカップを題材に研究を行うことは，本当に実世界で通用するロボットを実現するために，非常に有意義なことであると言える．

[*2]ボールはどのリーグにおいても，オレンジ色に着色されている．

1.3　従来研究

前節で説明したように，ロボットが実環境で知的に行動するためには，センサ情報の処理と行動決定の双方での工夫が必要である．センサ情報処理アルゴリズムの出力が行動決定アルゴリズムに入力されて，とるべき行動が決定されるわけであるが，この入出力される情報の性質の違いによって，行動決定アルゴリズムは大きく二つに大別できる．

そのひとつは，Fig.1.2(A) のように，センサ情報からロボットや環境の状態を表現するパラメータを推定するアルゴリズムが存在し，その出力から行動決定が行われるという方法である．もう一方は，Fig.1.2(B) のように，推定アルゴリズムが存在せず，センサの出力する数値が直接用いられて行動が決定される方法である．

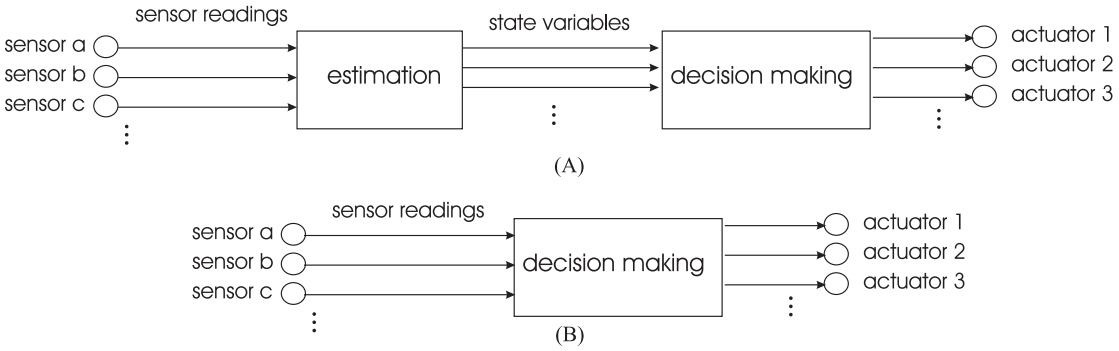

Fig. 1.2　行動決定法

4 足ロボットリーグでは推定の研究例として，自己位置推定が多く存在するが，自己位置推定を用いれば，上記の二つの方法のうち，前者を選択することとなる．つまり，センシングからは直接得られないようなロボットの位置，方向というパラメータ（内部状態変数）を設け，センサ情報からそれらを推定し，推定結果を用いて行動決定を行う．また，後者の例として，内部状態変数を用いず，画像処理から得られる物体の位置等の測定結果から，行動決定する手法も存在する．

本節では，まず推定アルゴリズムについて，自己位置推定に限定してロボカップやその他の移動ロボットで用いられている手法を説明する．次に行動決定法について，上記の双方の例について説明する．

1.3.1 自己位置推定法

4足ロボットリーグのフィールドでは，フィールドの周囲のランドマークと，ゴール，あるいはフィールド上のラインを用いて自己位置を求めることができる．初期の頃は，フィールド上のロボットの位置x, yと，方向θで構成される空間を格子状に区切り，全格子に対して，その格子中にロボットが存在する確率を求めるアルゴリズムがいくつか提案された [Veloso 98, Buschka 00]．Fig.1.3 にその一例を示す．これは，フィールドを格子状に区切ったものであり，色の濃さがロボットが存在する確率が高い部分を表している．なお，この図を引用した [Buschka 00] の文献では，これらの格子を用いて，ファジー推論によって自己位置を求めている．また，[Veloso 98] は，推定問題に用いられる基本的な手法であるベイズ推定を自己位置推定に利用したもので，車輪型ロボットを用いた Burgard らのアルゴリズム（Grid-based markov localization）[Burgard 96] を4足ロボットに応用したものである．

ロボットの移動領域を格子状に区切る方法は，ロボットの移動範囲が大きいと，計算量が大きくなる．それは，ロボカップのフィールド程度の広さ（4.2[m] × 2.7[m]）の場所でも問題となる．たとえば，2002年度のフィールドにおいてx, y, θ空間を，100[mm]×100[mm]×10[deg] の格子で区切ると，格子の数は 42×27×36=40824 個になり，この全てに対して実時間で確率計算することは，ERS-210 では現実的ではない．

そこで Burgard らは8分木によって，必要な部分だけ格子を細かくして自己位置推定を行うことを提案した [Burgard 98]．Fig.1.4 は，この手法を説明したもので，従来であると (A) のような格子数が必要であるのに対し，(B) のように確率の高い部分（色の濃い部分）のみを細かく区切ることで，メモリ使用量や計算量を抑制することができる．さらに，この方法では，格子の細かさに制限がないため，通常の格子のような精度の限界がないという利点もある．ただしアルゴリズムはかなり複雑になる．

Dellaert，Fox らは，[Burgard 98] と同等以上の効果が得られ，しかも非常に簡単なアルゴリズム（Monte carlo localization，MCL）を提案した [Dellaert 99, Fox 99]．これは，モンテカルロ法を用いた方法で，格子の代わりに Fig.1.5 のように移動空間中に多数の点（サンプル）を配置し，各サンプルに確率を持たせて，ロボットの存在確率を表現するものである．MCL は，ロボットの移動に合わせてサンプルを移動させ，センサ情報が入るとベイズ推定によりサンプルの持つ確率を操作し

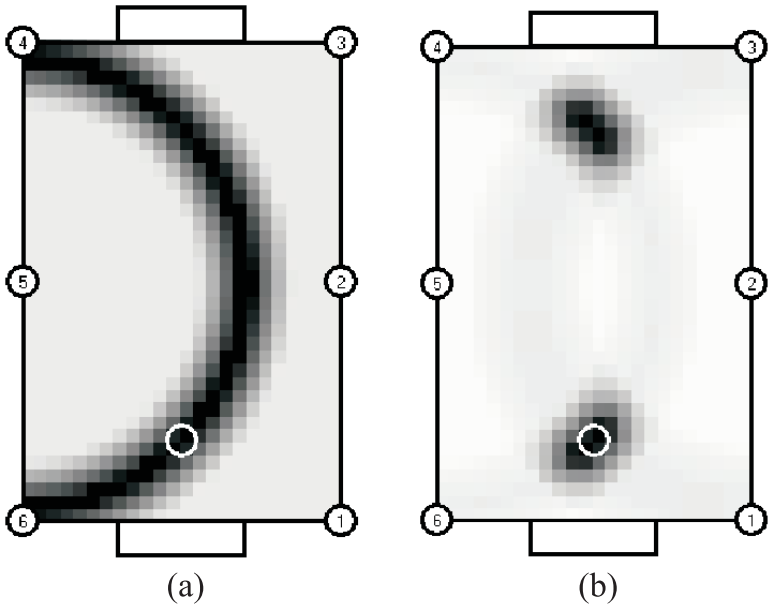

Fig. 1.3 格子状に空間を区切る自己位置推定法（[Buschka 00] より）

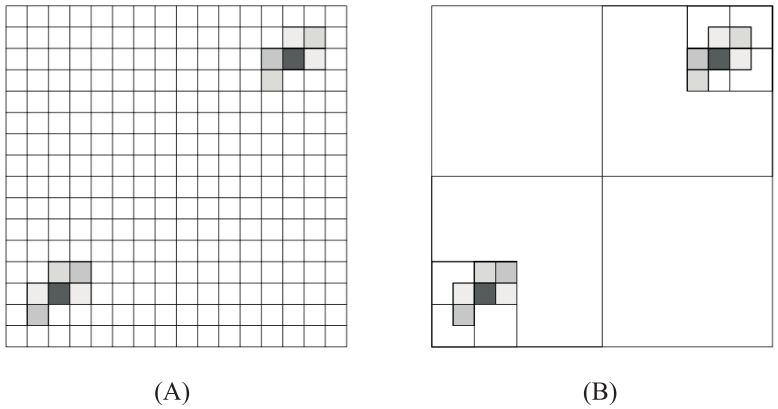

Fig. 1.4 8 分木を用いた空間の分割 ([Burgard 98] より)

て，ロボットの自己位置を推定する．サンプルは常にロボットの周囲に集まっているので，ロボットの移動する環境の広さは計算量に影響しない．ひとつのサンプルの操作にかかる計算量は，格子を用いる手法での一つの格子についての計算量と同程度なので，上記のいずれの問題も解決していると言える．

MCL は，ロボカップ中型機リーグや，4 足ロボットリーグで用いられた例がある [Lenser 00, Enderle 01, Gutmann 02]．これらの研究は，MCL を実機やロボカップのフィールドに適用する方法について扱っている．例えば Lenser らは，4 足ロ

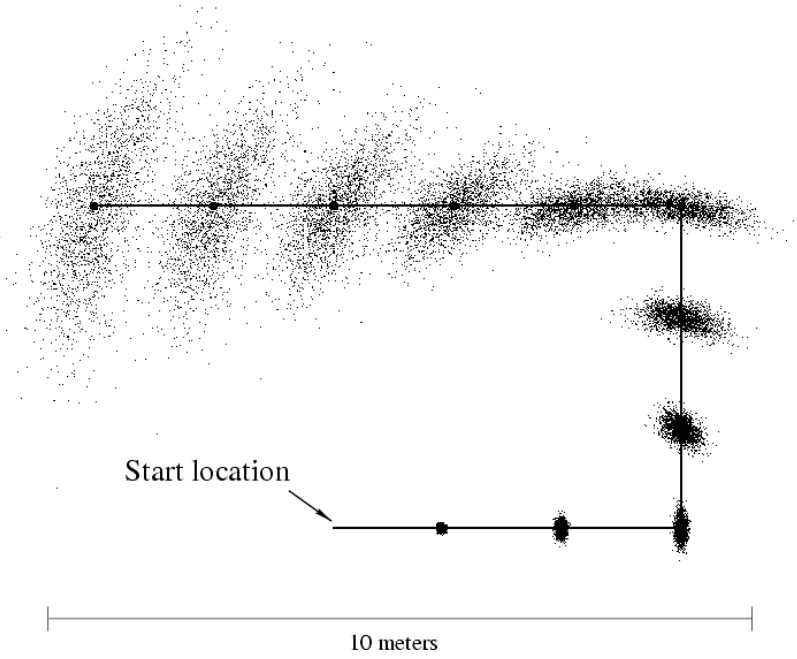

Fig. 1.5 サンプルによる存在確率の近似（[Fox 99] より）

ボットリーグのランドマークの位置計測を行うと，高頻度で大きな誤差を含む計測結果が出ることと，ロボットが実時間で計算できるサンプル数が数百個で，ロボットの自己位置を表現するには少なすぎることに焦点を当てて MCL を適用した．この例では，計測前のサンプルの分布と，計測によって得られる自己位置の情報が大きく矛盾するとき，計測前のサンプルの分布を無効にし，得られた計測結果で得られる自己位置の確率分布に従うようにサンプルを再配置するという方法をとった（Sensor Resetting Localization，SRL）．情報の矛盾は，センサ情報に大きな誤差が生じたときと，サンプルが不足して，本来の自己位置の確率分布を表現できないときに起こるので，この方法は，この二つを同時に解決することを意味する．ただしその代償として，矛盾後において，矛盾前の自己位置推定結果を全く用いないため，すこしロボット同士が接触したり，なんらかの許容範囲外のノイズがセンサに混入すると，最初から自己位置推定を行うこととなる．

ここで視点を変えて，推定結果を用いる側（行動決定アルゴリズム）から，従来の方法が持つ特性と，その問題を述べる．例えば，Fig.1.6 のように，ゴールにゴールキーパーロボットが，自己位置推定結果に基づいて，車庫入れの要領で後ろ向きに戻っていくタスクを考える．このタスクでは，キーパーが自己位置推定をある許容誤差以上に誤ると，Fig.1.6(A) のように，ゴールポストに衝突してしまう恐れがある．

タスクと，それにかける時間，許容される失敗確率の上限が決まっている場合には，推定アルゴリズムに許容精度を設定して，ロボットはそれを考慮してゴールポストから十分に距離を保ち，ゴール中央から戻るという行動設計が正しい方法である．このとき，ロボットには許容誤差範囲内に自己位置を一点だけ推定するだけの十分な時間が与えられることが通常である．一方，ロボカップのように非常に実時間性が求められる場合には，失敗確率には上限があるものの，それ以内であれば迅速に行動を行うことが要求される．Fig.1.6 のキーパーの例では，(B) のように，ポストに衝突することを覚悟して，余分な歩数をかけないでゴールに帰還する場合がある．

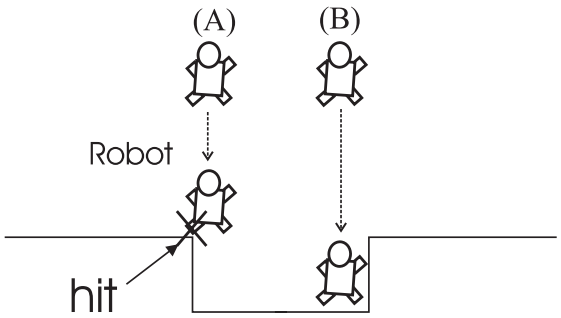

Fig. 1.6　ロボットの車庫入れタスク

これまで紹介してきた自己位置を確率的に表現する手法では，タスクが成功する領域と失敗する領域のロボットの存在確率から失敗確率が計算できる．つまり，許容誤差を定めてそれに基づいてロボットが移動するのではなく，ロボット自身が失敗確率を計算して，行動決定を行うことがこれらの手法では可能である．これによって，ロボットはタスクに必要な許容誤差を自身で計算しながら行動することになる．これは，設計者が与えた許容誤差をロボットが忠実に守りながら行動を行うよりも，柔軟であるといえる．

しかし，この確率計算を正確に行うには，自己位置推定アルゴリズムが自己位置の確率分布を正確に近似している必要がある．ERS-210 の CPU で自己位置を確率的に表現する場合，その表現力には限界がある．例えば SRL で扱えるサンプル数は，ERS-210 で数百程度であるが，これでは確率分布を表現できていない可能性がある．

また，入力されたセンサ情報に対して，どれだけの信頼を置いて自己位置を推

定するかということも，自己位置の確率分布を正しく推定するためには必要なことである．言い換えると，本来信頼性の低い情報を過信すると，自己位置の確率分布が余計に狭い分布となる．

カルマンフィルタ [加藤 87] を自己位置推定に用いる例は多いが，この手法では，この問題が顕著となる．この方法は，あるセンサ値がある平均値，分散に従う確率分布（計算の上では正規分布を用いる）に従ってばらつくことを前提してベイズ推定を行ったとき，計算が簡単な行列演算で表現できて，しかも推定結果が平均値と共分散で与えられるため，格子やサンプルを用いる手法よりも，格段に計算量では有利であり，出力の性質もパラメトリックで扱いやすい．

ところが，4 足ロボットリーグでは，カメラで観測対象を観測するとき，他機が対象を隠したり（オクルージョン），照明条件が変化して，観測対象の計測値に定常誤差が生じるという問題がある．これは，実世界における移動ロボットの場合でも，特に人などの往来が激しい環境や，窓から外の光が入ってくる室内環境，屋外で同様な問題に直面しうる．これらは，カルマンフィルタで用いられる「平均値，分散既知の正規分布」では表現できない誤差のばらつきを発生させる要因となる．カルマンフィルタは，このような誤差要因が発生しなければ，信頼できる推定値を得ることができる．しかし，定常誤差が起こると，定常的に推定結果の平均値は真の位置から外れ，その後立て続けに同じ性質の誤差が混入したセンサ値が入力されると，正規分布の数回の乗算によって，短時間で共分散の収束が起こる．そのため，カルマンフィルタは自己位置の確率分布を正しく推定するよりも，ある許容誤差内の自己位置を一点だけ求めることに適している．

ロボカップにおいて用いられているベイズ推定に基づく手法（格子を用いるもの，SRL，MCL）では，カルマンフィルタのように用いる確率分布が限定されていないため，定常誤差に対応することは可能である．このように，複数の確率分布モデルが使用できる手法は，マルチモーダルな自己位置推定と呼ばれる．しかし，4 足ロボットリーグのフィールドを，環境の変化が起こらないことを前提に，一点を推定することを目指して設計されているか，自己位置を確率的に表現するために，センサ値を入力する頻度を下げるなどの本質的ではない解決法を用いて設計されている．環境の変化を考慮しない方法では，事前に十分にセンサ値とロボット位置の関係について統計をとり，それを自己位置推定に反映させることで定常誤差に対処ができるが，このような事前計測を行わない（定常誤差に対応した）自己位置推定アルゴリズムを実装されたロボットの方が，より自律的で，より実世界向きであると言える．

1.3.2　行動決定手法

ロボカップでは，実時間性が要求されるため，オフライン計画や教示，学習を事前に行い，そこから得られた知識を用いてロボットが行動決定を行うという事例が多い．ここで言う知識は，センサ等からロボットが知覚するすべての状態に対して，どのような行動を選択するかを記述したデータベースである．単純な例として，Fig.1.7 に，2 次元で縦横斜め方向に動作可能なエージェントのためのデータベースを示す．エージェントは，自己位置 x, y を知覚（センサーから直接得ても，推定で得てもよい）できれば，なにも考えずにデータベースの矢印の方向に進めばゴールにたどり着くことができる．このようなデータベースを持ったロボットは，自身や環境の状態を推定するか，あるいはセンサ情報を直接利用して，データベースを参照して行動決定することで，行動決定に必要な計算量を極限まで減少させることができる．

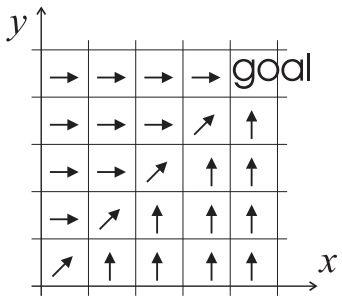

Fig. 1.7　行動決定のためのデータベース

ただし，センサの出力には誤差が含まれ，それを用いて推定をする場合には誤差や曖昧さが生じるので，ロボットがこの方法で移動することは簡単ではない．ここで誤差はセンサ値の誤差を推定で考慮しなかった結果発生し,「曖昧さ」とは，誤差を考慮した結果やセンシングの頻度不足によって残った，情報の不確定性のことである．つまり，誤差を推定で効果的に吸収しても曖昧さが残るため，ロボットは曖昧さを考慮して行動する必要がある．前述の「失敗確率」の計算も，この曖昧さの考慮の一部分である．

したがって，データベースを作成する段階か，オンラインで用いる段階のどちらかで，曖昧さの考慮が行なわれる必要があり，これは実際に研究対象となって

きたことでもある．ここでは，データベースを作成してそれをオンラインで用いる方法の例を説明し，それぞれについて，センサ情報の曖昧さに対する特徴を議論する．

まず最初に，Fig.1.2(B) のように，センサ情報（あるいは，簡単な情報処理の出力）をそのまま状態変数として扱い，強化学習 [Sutton 98] や教示を用いて，行動と直接結びつける方法を挙げる．学習や教示などでは，あるセンサ情報に対して，それにどれだけ定常的に誤差があるか，どれだけばらつくかは問題とせず，ただその情報が得られたときに，ある行動を選択した結果が良かったかどうかを問題とする．このような考え方でデータベースを作成する方法には，以下のような例がある．

光永らは，ロボカップ 4 足ロボットリーグの環境において，カメラに写ったランドマークやボールの位置と，観測行動，移動行動を，教示によって結び付け，ロボットに，ボールをゴールまで運ぶ行動を獲得させている [光永 01]．

また，高橋らは，車輪型ロボットに，サッカーにおけるシュート行動やボール追従行動を獲得させるために，強化学習を用いてセンサ情報（単純な画像処理結果）と行動（モータの出力）を結びつける研究を行なっている [Takahashi 99, 高橋 99]．

これらの例では，センサ情報の持つ定常的な誤差は全く問題にならない．また，センサ情報がばらついても，教示量や学習の経験量が多ければ，そのばらつきのもとでタスク達成の期待値が最も大きくなる行動を得ることが可能である．しかしながら，これらの手法では，もしセンサ情報の特性が変わったときに，即座に対応ができない．また，上記の二例ではカメラを用いているため，照明条件の変化に非常に弱いことが予想される．この場合，教示であれば，再度行なうことになる．強化学習においては，ロボットが実際にタスクを行ないながら，変化した照明条件に適応して，データベースを改善することはできる．しかしこの改善を，ほんの数秒で行なうことは不可能であり，かなりの試行回数，実機を用いた再学習，再教示を行う必要がある．

次に Fig.1.2(A) の方法についての研究例を挙げる．この方法では，センサ情報を直接使用しないため，実機を用いないで計画問題を解くことができる．例えば，移動ロボットの経路計画問題は，環境の地図とロボットの位置だけを用いて計算機中で解くことができる．また，自己位置推定で挙げた研究例のように，推定の

曖昧さの表現も出力できる推定アルゴリズムを用いる場合には，その表現を有効に利用できる計画方法や，オンライン計算が必要となる．

まず，ロボットの行動を，情報が曖昧であることを最初から考慮して計画するものを挙げる．代表的なものは，計画問題を解く状態空間を，情報の曖昧さを表す次元を加えて拡張する方法である．この拡張された空間は，[Barraquand 95, LaValle 00] において情報空間と呼ばれている．情報空間の考え方は，動的計画法 [Bellman 57] を用いた解法とともにいくつか用いられた例がある．

Roy らはソナーを持つロボットの経路計画問題を，ロボットの位置二次元，方向一次元のほかに，Fig.1.8(a) のように，自己位置の曖昧さを表すエントロピーの一次元を加えた，四次元の情報空間を構成して解いた [Roy 99]．得られる解はソナーから情報が得られる壁際を，ロボットがなぞるように移動するというものであることから，この方法は Coastal Navigation と呼ばれている．ただし，この手法では，計画の時点では曖昧さを一次元で表現しているため，曖昧さの度合いは分かるが，それが状態空間上でどのように曖昧なのかを知ることはできない．

一方，深瀬らの研究では，自己位置がどのように曖昧であるかを考慮して計画するために，筆者らの自己位置推定アルゴリズム（Uniform Monte Carlo Localization, Uniform MCL）[Ueda 02]（第 3 章で詳しく説明する.）の出力（確率分布）のうち，典型的なものをサンプリングして，サンプリングした分布形状を状態変数の一つとして情報空間を構成した [深瀬 02]．つまり，本来無限次元である確率分布を有限個に分類し，Fig.1.8(b) のように一軸上に並べることで，情報空間を作成した．

これら二つの例は，どちらもどの場所で，どれだけ自己位置の情報が得られるかの確率を，実機を用いて事前に計測している．そして，得られた確率モデルを動的計画法に反映させて解を得ている．したがって，動的計画法で得られた解は，確率モデルを計測した状況が変化すると最適性が保証されなくなる．ロボカップの場合には，前述のように照明条件が変化するので，最適性の保証はない．

情報空間の考え方で，学習を用いて行動獲得する方法も存在する．観測によって，現在の状態が部分的にしか分からない場合の行動決定問題は POMDP（Partially Observable Markov Decision Process）問題と呼ばれ，得られる情報が曖昧さを持つ場合の問題も，この範疇に入る．例えば，Thrun は，MCL で推定された状態（これは，サンプルの分布形状で与えられる）ごとに，強化学習を用いて最適な行動

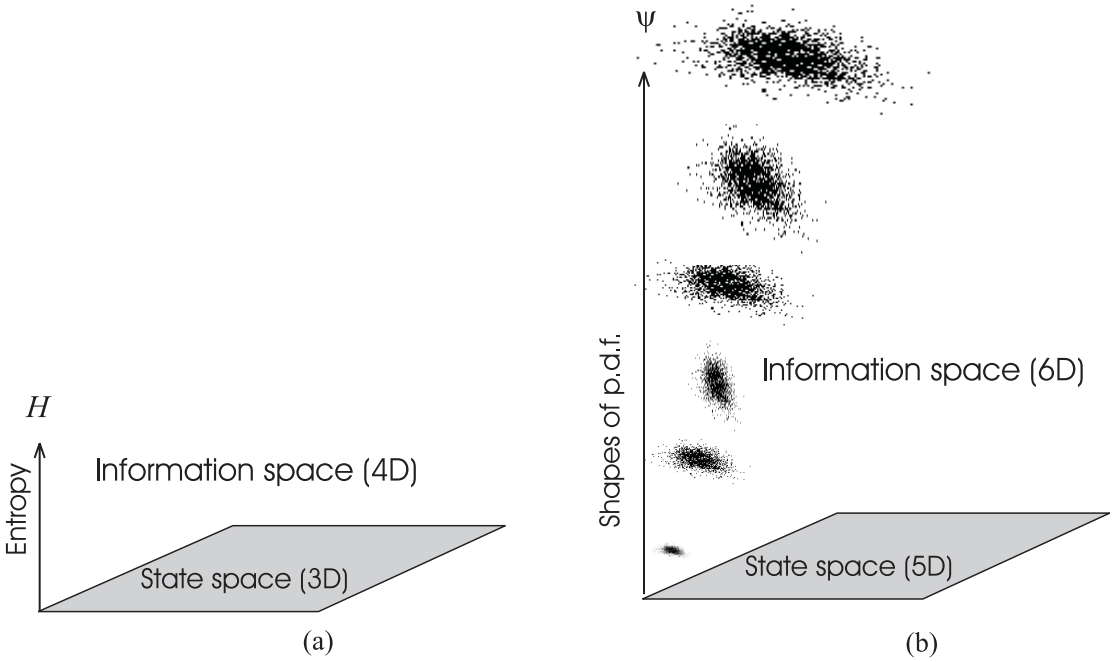

Fig. 1.8 情報空間の例．(a) が [Roy 99]，(b) が [深瀬 02] のもの．

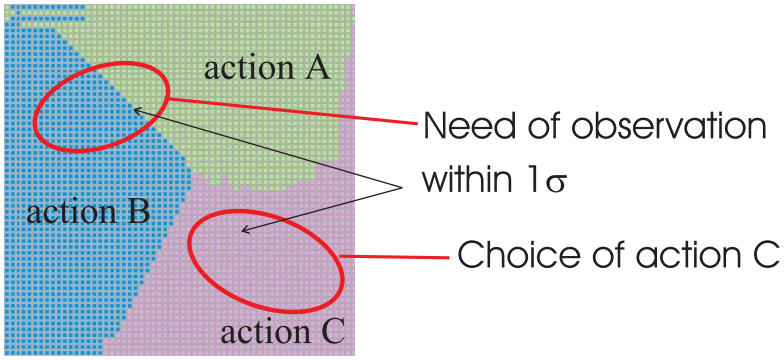

Fig. 1.9 [横井 01] での行動決定方法

を割り当てることを提案している（Monte Carlo POMDP）[Thrun 00a]．これは，[深瀬 02] では事前に与えられていた分布形状を学習中に獲得できるアルゴリズムであり，環境の変化にも順応できる可能性がある．

情報空間を扱う場合は，どうしても空間が大きくなってしまう．大きな空間を扱うための研究は，現在の強化学習の研究で中心となっており [Takahashi 99, Tsitsiklis 96, Tuyls 02]，未だ根本的な解決には至っていない．一方，ロボカップのような小規模な環境でさえ，考慮できる状態変数は数個程度で，試合全体を表現する状態変

数のうちのほんの一部である．よって，ロボカップへの適用に話題を限定すると，曖昧さを表現して次元を増やすよりも，他の状態変数を考慮して，より複雑な状況を考慮したデータベースを構築した方がよいと言える．

曖昧さを考慮しないで得られた動的計画法の解を用いて，曖昧さを考慮した準最適な行動を得る方法もある．[横井 01] では，脚型のサッカーロボットが，ある決められた地点で，決められた向きで止まるための行動決定法を，次のようなオフライン計算，オンライン計算によって設計した．

オフライン計算 計算機を用いて，曖昧さを考慮しない行動計画を動的計画法で解き，ロボットの位置，方向をロボットの取るべき歩行行動に変換する方策を得る．

オンライン計算 SRL の結果から誤差楕円を作成し，誤差楕円を Fig.1.9 のように方策に当てはめる．楕円内がすべて同じ歩行行動であったら歩行，二種類以上の行動があったら自己位置推定のために，立ち止まって観測を行う．

この方法は，観測の確率モデルを用いないので，先ほど述べたような特定のフィールド，環境への依存性がない[*3]．しかも，計画に必要な計算量やメモリの量も，情報空間を構成する方法に比べて非常に少なくてすむ．しかしながら，どれを選択しても違いがない複数の歩行行動がある場合にも観測が行われ，これは無駄であるので，改善が必要である．

以上の議論をまとめて，用いた評価指標に対する各アルゴリズムの性能を表にすると，Table 1.1 のようになる．この表で大事なことは，曖昧さへの考慮という点では，どの手法も対応しているが，その曖昧さの原因となる環境変化に対して，即時対応する手法が [横井 01] 以外にないことである．また，情報空間で計画，学習する方法は，メモリ使用量が多いにもかかわらず，環境が変化すると最適性が失われるという問題が共通していることも，この表から分かる．

[*3]ただし，用いていた SRL が依存しており，結果として環境の変化に弱くなっている．

Table 1.1 各行動決定法の比較

	学習	**教示**	**CN**	**MC**	**[深瀬 02]**	**[横井 01]**
情報の曖昧さへの対応	○	○	△	○	○	○
環境変化に即時対応	×	×	×	×	×	○
オフラインメモリ使用量	-	○	△	-	×	○
オフライン計算量+人間の作業量	-	△	△	-	△	○
オンラインメモリ使用量	○	○	△	×	×	○
オンライン計算量	○	○	○	×	○	○
複数の行動候補からの選択	○	-	-	○	-	×

CN: Coastal navigation, MC: Monte Carlo POMDP.

○: 対応可能，△: 対応可能（○より劣る），×: 対応不可能，-: オフライン計算なし，あるいは複数の行動候補から選択する必要がない.

1.4　研究の目的

以上の議論から，本研究では，

環境変化に即時に対応する
自律移動ロボットの実時間行動決定の実現

を目標として，

(1) **自己位置推定アルゴリズム**

(2) **行動決定法アルゴリズム**

と，その他必要なアルゴリズムを開発し，検討する．上記のアルゴリズムは，以下の問題に対処する必要がある．

- **自己位置推定について**
  - 実時間性
  - 環境の変化（照明条件の変化等）を考慮した推定
  - 曖昧さの表現
- **行動決定法について**
  - 推定アルゴリズムの曖昧な出力を有効利用
  - 計算資源の削減，計算時間の短縮（オフライン時）
  - 計算資源の削減，実時間性（オンライン時）

(1) については，実時間性のため，従来とは異なったアプローチで MCL を実装する．また，環境の変化に対応し，曖昧さを忠実に表現するため，MCL のようなベイズ推定がロボットの自己位置推定に適用される場合，何が問題となってロバスト性が損なわれたり，曖昧に表現すべき場合に断定的な推定をしてしまうのかを考察し，その考察に基づいて自己位置推定アルゴリズムを設計する．

(2) については，[横井 01] の手法を改善したアルゴリズムを設計，実装する．設計する手法は，次のようなオフライン計算と，オンライン計算で構成する．

オフライン計算 計算機を用いて，曖昧さを考慮しない状態空間において，動的計画法の価値反復 [Sutton 98] により，行動を計画する．

オンライン計算 オフライン計画で得られた状態価値関数（ある状態の価値）と，方策（ある状態でとるべき行動が記述されたデータベース）をロボットに実装する．曖昧な状態推定の出力と状態価値関数から，現在の状態価値の期待値と，行動後の価値の期待値を比較し，最も価値が増加する行動を選択する．

本研究では，自己位置推定が重要な，キーパーの役割のロボットに設計したアルゴリズムを実装し，その有効性を評価する．また，実機において評価しにくいことについては，当研究チームで開発されているシミュレータを用いて評価を行う．

1.5　本論文の構成

本論文の構成は，次のようになっている．また，構成をまとめたフローチャートを Fig.1.10 に示す．

2 章では，ロボットの行うタスクを示し，そのタスクの定式化を行う．

3 章では，状態推定の例として，自己位置推定問題を扱い，自己位置推定アルゴリズムを設計してロボットに実装する．

4 章では，動的計画法によってロボットの行動を設計する手順を説明し，例としてフォワードとキーパの行動計画を行う．

5 章では，3 章の状態推定アルゴリズムと，4 章での行動計画の出力から，状態推定が曖昧な状態で行動を決定するアルゴリズムを設計する．

6 章ではシミュレーション，7 章では実機実験によって，設計，実装した自己位置推定アルゴリズム，行動決定アルゴリズムを評価する．

8 章では，本論文での結論と今後の課題について述べる．

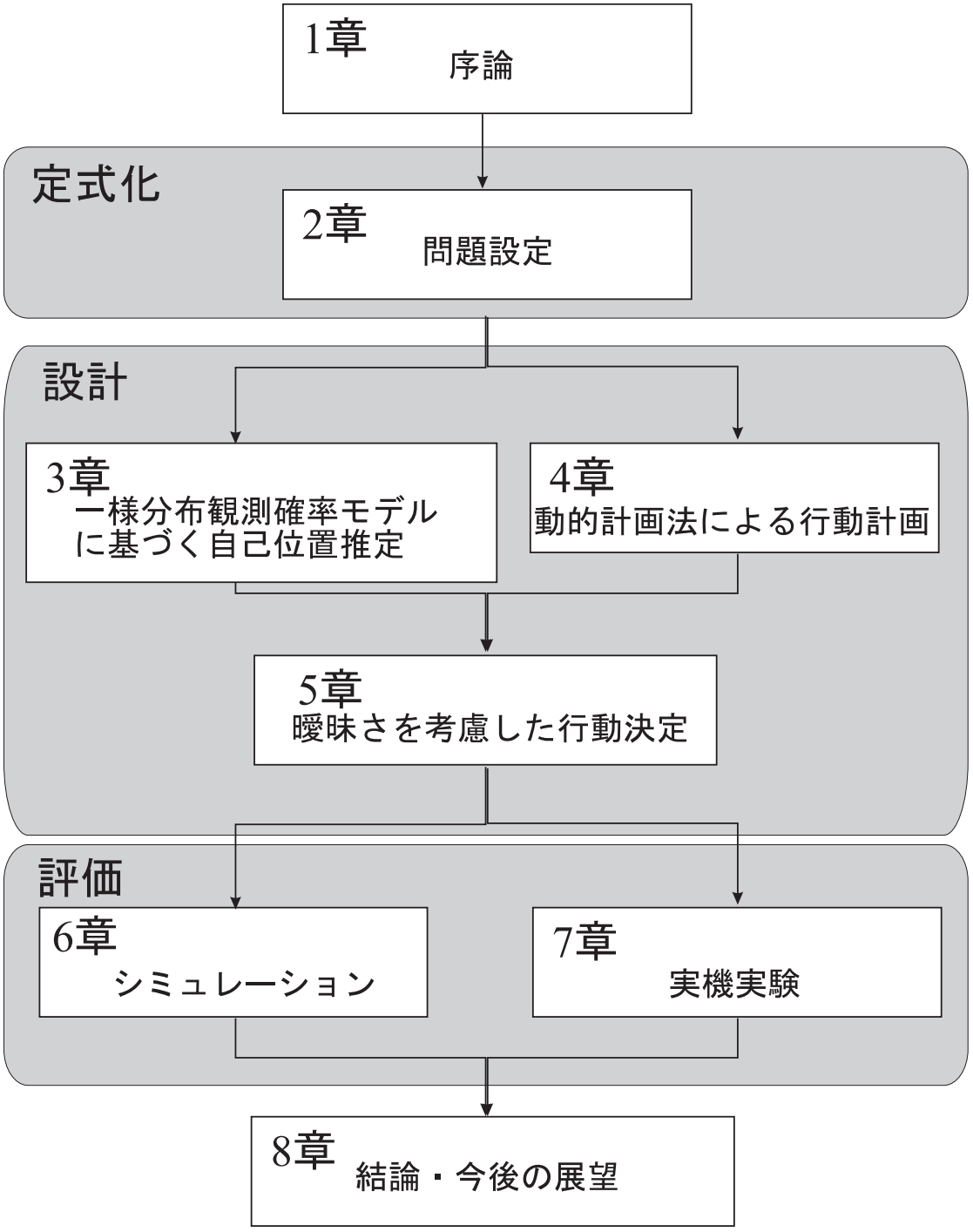

Fig. 1.10　本論文の構成

第2章　問題設定

2.1　はじめに

本章では，本研究で用いるフィールド，ロボットについて説明し，ロボットが行うタスクについて説明する．

2.2 節では，ロボットが行動を行う環境である，サッカーフィールドの説明を行う．フィールドそのものや，その周囲の物体の寸法や形状について述べる．そして，本研究で扱う環境変化を定義する．

2.3 節では，ロボットの自由度，センサ（CMOS カメラ），計算資源を説明する．また，タスクの遂行のために用いる動作について定義する．

2.4 節では，タスクの説明を行い，次章以降の議論のために定式化する．

2.5 節では，本章のまとめを行う．

2.2 フィールド

2.2.1 各物体の寸法

フィールドの大きさは，幅 2700[mm]，ゴール間の距離が 4200[mm] である．また，ゴールは，入り口の幅が 600[mm]，奥行きが 350[mm] の長方形の領域になっており，ここにもロボットは入ることができる．フィールド内の物体は，Fig.2.1 のように，色で識別できるようになっている．参考のため，フィールドの詳細な寸法を Fig.2.2 に示す．

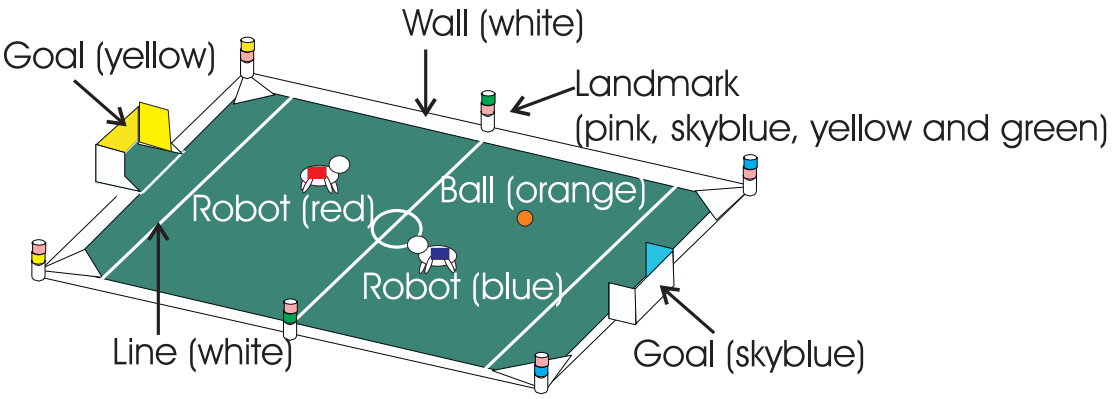

Fig. 2.1 2002 年度フィールドの概観

Fig.2.1 中の本研究に関連する物体について説明する．

ゴール 水色あるいは黄色に塗られた高さ 300[mm] の鉛直な板で囲まれている．Fig.2.3 に，ゴール板の詳細な寸法を示す．

ランドマーク 2 色に塗り分けられた円柱である．Fig.2.4 に，このランドマークの寸法を示す．2 色の違いにより，6 個のランドマークは互いに識別可能である．ランドマークの座標は，Table 2.1 のように表せる．

ボール 直径約 80[mm] のオレンジ色の球である．

壁 フィールドの周囲には，傾斜が 45[deg] の壁があり，ボールやロボットがフィールド外に出ないようになっている．また，四隅には，ボールがロボットに閉じ込められるのを防ぐために，三角形の緩い傾斜がある．

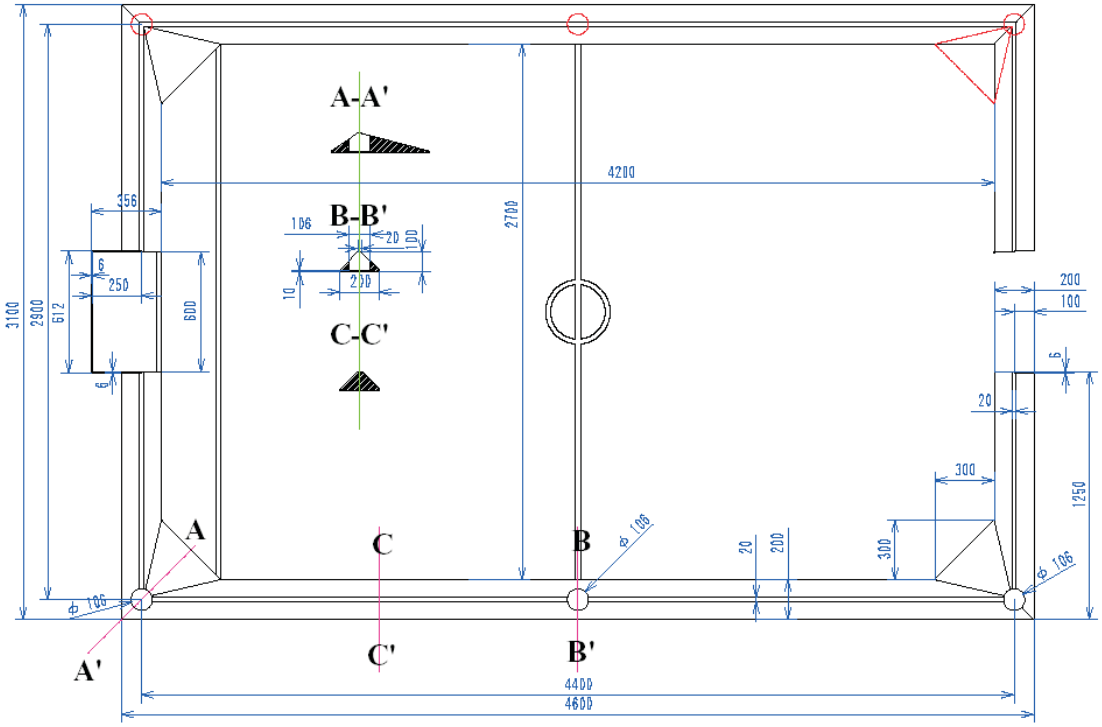

Fig. 2.2 フィールドの寸法

Table 2.1 ランドマークの位置と色

x[mm]	y[mm]	上部の色 (Fig.2.4 の A)	下部の色 (Fig.2.4 の B)
-2200	-1450	pink	yellow
0	-1450	pink	green
2200	-1450	pink	skyblue
-2200	1450	yellow	pink
0	1450	green	pink
2200	1450	skyblue	pink

2.2.2 環境変化

本研究の目的は，環境の変化に対応した行動決定であるが，ここで，本研究で扱う「環境変化」を具体的に定める．本研究では，センシングによって得られる情報の変質がおこるような状況を，環境の変化と呼ぶ．

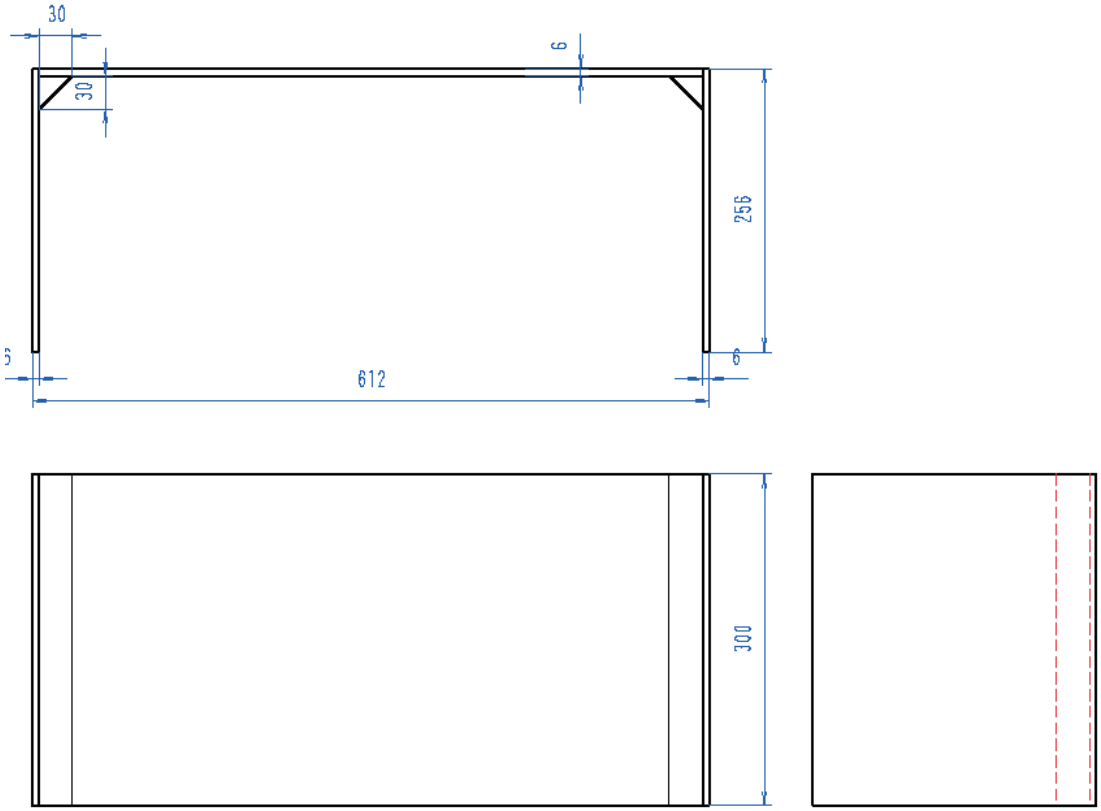

Fig. 2.3　ゴールの寸法

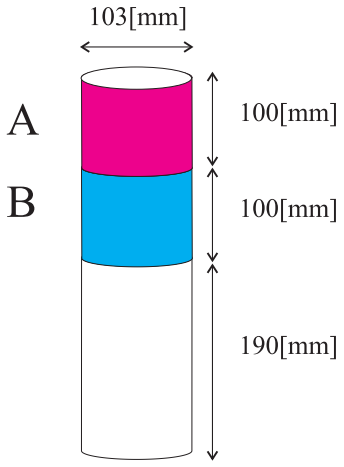

Fig. 2.4　ランドマーク

環境の変化は，様々な要因から起こるものであるが，試合に関係することを解決するために，

- フィールドの照明条件が変化して，カメラ画像の明るさが変化する．
- 本来あるべきランドマークやゴールが，存在しない．

という問題に集中して，アルゴリズムを設計，実装する．前者は，実際に試合で起こりうることである．後者については，試合中は起こらないが，代わりに他機がこれらを視界から隠すなどの理由で，似たような状況が起こりうる．

他にも，ランドマークの位置が大きく移動したり，ゴールの位置が変わったりなどと，ということも環境の変化として考えられる．これらは，サッカーの試合に関してはありえないが，推定を行うための観測対象の位置が変わることや，タスクの質が変化する課題として挙げた．このような課題も，今後取り上げられるべきであるが，これらについては，本論文では扱わないこととする．

2.3 ロボット

2.3.1 センシング

本研究で用いるロボットの外界センサは，カラー CMOS カメラのみである．この CMOS カメラの画角，画素数等は，以下の表のとおりである．CMOS カメラが

Table 2.2 CMOS カメラの仕様（[OPE 02] より）

画素数（ロボットの RAM に送るもの）	幅 168× 高さ 144[pixel]
画角	水平方向 57.6× 垂直方向 47.8[deg]
カラーの表現形式	YCbCr 形式（各 256 階調）
フレームレート	40[ms]

搭載されている頭部の関節は 3 自由度で，それぞれがカメラのチルト角，パン角，ロール角を制御するため関節に相当する．それぞれの関節の可動範囲を，Fig.2.5 に示す．また，画像処理の説明のために，この図のようにカメラ座標系 Σ_{cam} を設定する．

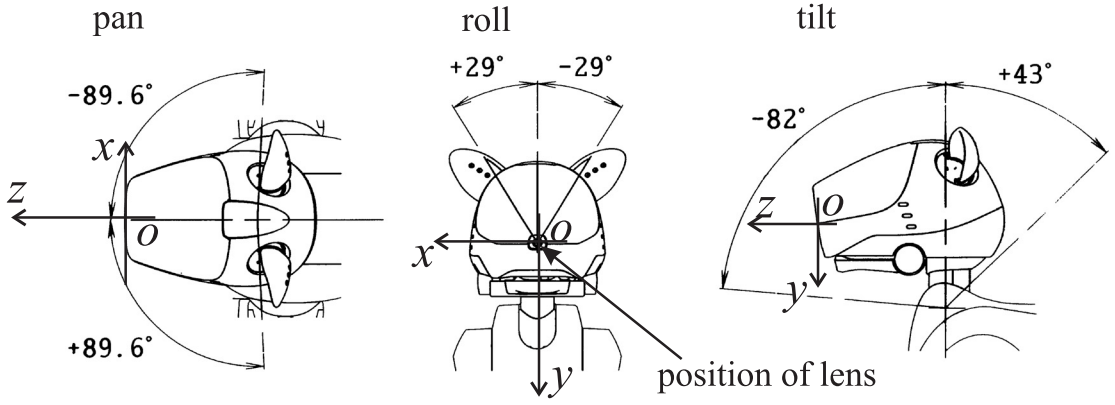

Fig. 2.5 頭部関節の可動範囲とカメラ座標系（[OPE 02] より）

フィールドは，前節の説明のように，色で物体が識別できるようになっている．当研究グループでは，YCbCr 値を各 256 階調から 128 階調に落として，各 YCbCr 値の組み合わせに対して赤や青の「色の呼び名」を指定した 3 次元配列（色識別テーブル）を事前に作成して，ロボットに実装している．そして，YCbCr 画像がカメラから得られると，すぐに色識別テーブルから各画素の色を識別して，「色の呼び名」が記述されている画像（色抽出画像）に変換し，それを用いて画像処理

を行っている．Fig.2.6 にその例を示す．この図の (a) がカメラ画像で，この画像から (b) のように，各色が抽出された画像が作成される．

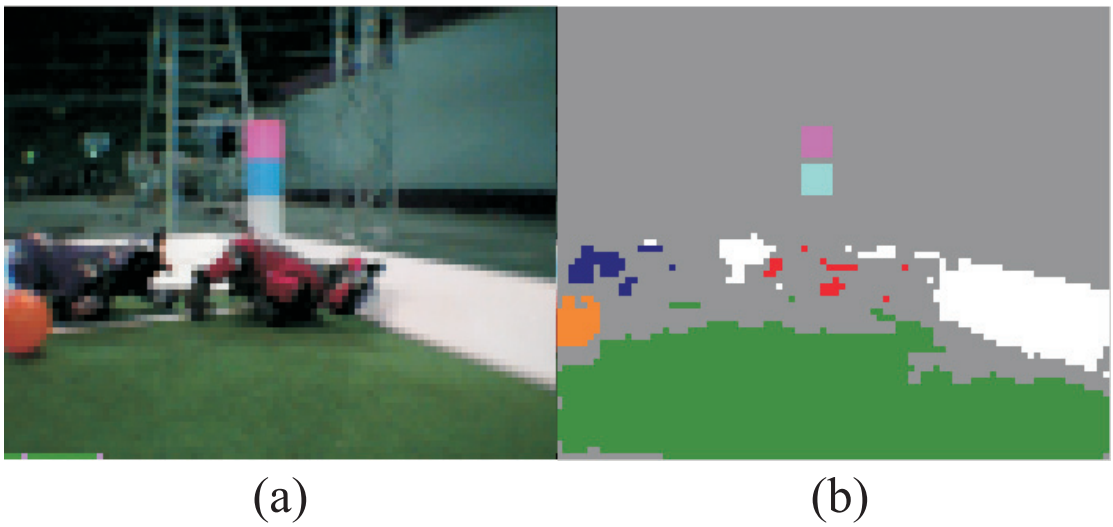

Fig. 2.6 CMOS カメラの画像 (a) と，それを色に変換した画像 (b).

2.3.2 行動要素

ロボットは，40 種類の歩行コマンドによって制御される．これらの歩行コマンドによる移動量は固定されている．Fig.2.7 のように，ロボットの前方に x 軸，左方に y 軸を定めてロボット座標系を定義する．このとき，この座標系で表される各行動後のロボットの位置，向きを 10 回程度計測して平均をとると，Table 2.3 のようになる．移動量は，歩容の再現性の問題や，床面の摩擦によってコマンドごとにばらつくが，これは 1-2[mm]，1-2[deg] 程度のわずかなものであるため，特に記載はしない．

ロボットは，これらの歩行コマンドを，すべて同じ時間（0.7[s]）で実行する．したがって，移動に関するタスクにおいては，最短時間で実行することと，最少歩数で実行することは，同義となる．また，行動の命令の他に，歩行を停止させることもできる．

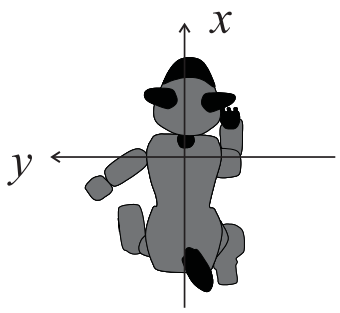

Fig. 2.7 ロボット座標系 Σ_{robot} (top view)

Table 2.3 歩行動作の移動量一覧

Name	x[mm]	y[mm]	θ[deg]	Name	x[mm]	y[mm]	θ[deg]
WalkForward	113	0	0.0	RightForward15	84	-16	2.0
WalkBackward	-87	0	0.0	LeftForward15	102	14	-2.6
HalfTurnR	5	-20	-32.5	RightForward30	74	-36	0.0
HalfTurnL	5	15	28.0	LeftForward30	102	32	-4.4
RightForward	75	-77	0.0	RightForward60	48	-102	0.0
LeftForward	72	79	0.0	LeftForward60	46	102	0.0
RightSide	0	-114	0.0	RightForward75	34	-116	0.0
LeftSide	0	107	0.0	LeftForward75	26	121	0.0
RollRight	35	-30	17.0	RightBackward15	-94	-36	0.0
RollLeft	35	30	-30.0	LeftBackward15	-95	35	0.0
DribFF	66	0	0.0	RightBackward30	-86	-56	0.0
DribBack	-46	0	0.0	LeftBackward30	-84	55	0.0
DribRF	55	-58	0.0	RightBackward60	-40	-110	0.0
DribLF	56	65	0.0	LeftBackward60	-42	106	-4.0
DribRightSide	0	-80	0.0	RightBackward75	-26	-115	0.0
DribLeftSide	0	66	0.0	LeftBackward75	-23	95	-5.0
DribRollRight	30	-25	18.0	RightForwardTurnLeft	50	-30	17.0
DribRollLeft	30	25	-30.0	LeftForwardTurnRight	70	25	-15.0
RightBackward	-83	-84	-2.8	RightForwardTurnRight	80	-92	-15.0
LeftBackward	-69	73	2.0	LeftForwardTurnLeft	80	100	15.0

2.4 タスクの設定と定式化

本研究では，Fig.2.1 に見られるロボカップ 4 足ロボットリーグのフィールドにおいて，キーパーロボットの移動に関するタスク（ゴール内での待機，ボールへの接近等）を，少ない歩数で効率よく行うための行動決定について，提案する手法を適用する．以後の議論のため，必要な記号の定義を行う．

まず，フィールド座標系 Σ_{field} を Fig.2.8 のように，フィールド中央を原点として，センターライン上に y 軸，それと垂直に x 軸を設定する．ロボットの位置は，この図のように (x, y, θ) で表現される．また，ボールの位置は，Fig.2.9 のように，ロボットからの相対位置の極座標 (r, φ) で表現する．ロボットが一台，ボールが一個の環境を想定し，これらの 5 つのパラメータ（状態変数）で，フィールド上の状況を表現する．これらのパラメータで構成される空間を，状態空間 $\mathcal{X}$ で表現する．

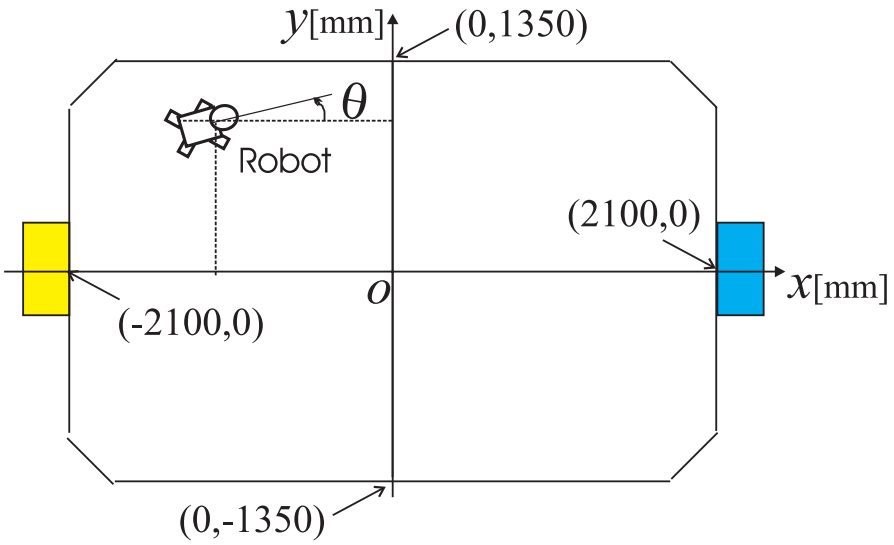

Fig. 2.8 フィールド座標系 Σ_{field}

本論文で設計，実装されるアルゴリズムは，この状態変数を推定，計測するまでの部分と，それらの結果から行動決定する部分に大別できる．それぞれの過程について説明に必要な記号を定義し，定式化を行う．

状態変数の計測，推定 状態変数 x, y, θ, r, φ で構成される状態空間 $\mathcal{X}$ 中の現在の状態ベクトル $\boldsymbol{x}$ を，センシング等によって求める．真の状態ベクトル $\boldsymbol{x}$ を一点に

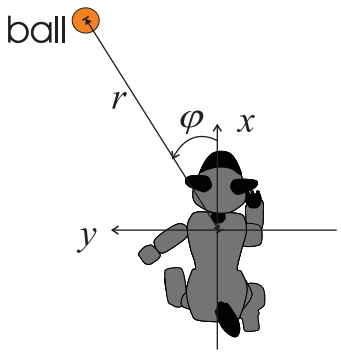

Fig. 2.9　ボール位置の表現

断定することは不可能なので，なんらかの曖昧さの表現を伴って推定されるべきである．ではあるが，本論文ではボールの位置 (r, φ) を計測するときは，一つの画像中のボールの位置から断定的に求めることとする．これは，既存のボール位置計測アルゴリズムを利用するという理由と，ボールの位置は，ロボットから遠い場合に精度が重要でなくなり，逆に近い場合には精度よく位置が求められるため，曖昧さを表現しなくても，タスクに支障がないという理由からである．ただし，これは本研究で取り上げられるタスクについてのみに言えることである．

自己位置に関する状態変数 x, y, θ については，曖昧さを考慮して推定を行う．この推定の出力は，確率密度関数 $bel(x, y, \theta)$ で表現される．この推定に使用できる情報は，カメラ画像と，デッドレコニングであるとする．次章以降では $bel(x, y, \theta)$ を，$xy\theta$ 空間中の点を ℓ として $bel(\ell)$ と表現する．また，状態ベクトル $\boldsymbol{x} = (x, y, \theta, r, \varphi)$ に対して，現在の状態が $\boldsymbol{x}$ である確率密度関数を $bel(\boldsymbol{x})$ と表現する．ただし，r, φ は確率的に推定されないため，これは説明のために抽象的に用いられる．

行動決定　状態推定結果を用いて，40 種類の歩行コマンド集合 $\mathcal{A} = \{a_1, a_2, \ldots, a_{40}\}$ から，適切な行動コマンドをひとつ選択し，実行する．この選択のための，決定論的状態 $\boldsymbol{x}$ から行動コマンド a への写像を，

$$\pi : \mathcal{X} \mapsto \mathcal{A} \tag{2.4.1}$$

と，π で表現する．この写像 π を方策と呼ぶ．序論で述べたように，これはオフライン計画で求められる．

ただし，状態推定結果は，確率密度関数 $bel(\boldsymbol{x})$ で表現されるので，実際にロボッ

トの行う行動決定は，このような状態がひとつに断定できるときの方策 π ではなく，確率密度関数 $bel(\boldsymbol{x})$ から行動コマンド a への写像を用いたものとなる．確率密度関数全体の集合を Ω で表すと，ロボットは，ある確率密度関数から行動コマンドへの写像

$$\Pi : \Omega \mapsto \mathcal{A} \tag{2.4.2}$$

に基づいて行動決定を行うこととなる．Ω を状態空間とみなすと，Π も方策と呼べる．

これまの定式化や記号類をまとめると，Fig.2.10 のようになる．推定の部分では，(x, y, θ) が確率的に表現されて，ボールに関する計測値とあわせられて $bel(\boldsymbol{x})$ が出力される．一方，オフライン計算によって，決定論的に方策 π が求められるため，入力は，$bel(\boldsymbol{x})$ ではなくて $\boldsymbol{x}$ となる．したがって，オフライン計画によって得た π か，あるいは別のものを用いて，この入出力の不整合を解消する方法が必要となる．具体的な例として最も簡単な方法を挙げると，$bel(\boldsymbol{x})$ の各状態変数の平均値で構成される状態ベクトル $\bar{\boldsymbol{x}}$ を求めて，これを入力する方法がある．しかし，これでは曖昧さの考慮という目的を達成できないので，より工夫した方法が必要である．

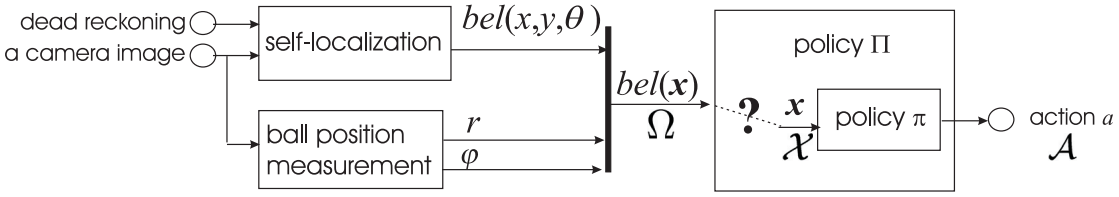

Fig. 2.10　定義した記号・オフラインとオンライン間の不整合

以上の方針に従って，次章以降で設計，実装が行われるアルゴリズムの入出力関係を簡潔に表現すると，

- 第 3 章：画像，デッドレコニング $\longrightarrow bel(x, y, \theta)$
- 第 4 章：$\boldsymbol{x} \in \mathcal{X} \longrightarrow a \in \mathcal{A}$
- 第 5 章：$bel(\boldsymbol{x}) \in \Omega \longrightarrow$ オフライン計画結果，何らかの方法 $\longrightarrow a \in \mathcal{A}$

となる．

ここで行った定式化では，状態が既知のとき，状態はマルコフ的で，マルコフ決定過程（Markov decision process, MDP）[北川 67] によって行動決定が定式化できることを前提にした．ある時刻 t における状態 $\boldsymbol{x}_t$ があったとき，その次の状態 $\boldsymbol{x}_{t+1}$ を予測するために過去の状態 $\boldsymbol{x}_{t-1}, \boldsymbol{x}_{t-2}, \ldots$ を知る必要がないとき，この状態はマルコフ的であると呼ばれる．任意の状態がマルコフ的である状態空間を構成できるタスクに対する行動決定過程はマルコフ決定過程と呼ばれ，現在の状態から，一意にとるべき行動が決定できる．

もちろん，ボールが速度を持って動き，他機が存在する限り，サッカーは，自己とボールの位置で構成される状態空間 $\mathcal{X}$ ではマルコフ的ではない．しかし，これらのことを考慮して計画するには，ボールの速度や他機の位置，速度を正確に（あるいは確率的に）計測できる手法が必要であり，このような手法が 4 足ロボットリーグに存在しないので，本論文ではこれらのパラメータについては扱わない．また，状態が自機の行動とは無関係に変化しても，変化後の状態に対して瞬時に行動決定することで，ある程度の対応は可能である．

2.5　おわりに

本章では，まず最初に，序論で目標と定めたアルゴリズムを具体的に適用する環境，ロボットの説明を行った．ロボットは，フィールド（4.2×2.7[m]）上の数十から数百 mm の物体の映った画像を画素数 176×144 の CMOS カメラから得て，それを色抽出画像に変換し，それを用いて環境の情報を得る．また，ロボットは，移動量固定の 40 種類の行動要素から行動を選択することで，移動を行う．

次に，問題を定式化し，設計，実装する各アルゴリズムの入出力関係を説明した．ロボットは画像処理からボールの位置を計測し，画像処理とデッドレコニングから自己位置を推定する．計測，推定結果から，オフライン計画の結果を用いて行動が決定されるが，確率的表現で推定された状態を，オフライン計画で求めた方策に入力することはできない．そこで，確率的表現を用いたオフライン計画結果の利用方法を設計，実装する必要がある．したがって，次章以降で，この方法が，状態推定，オフライン計画とともに設計，実装される．

第3章　一様分布観測確率モデルに基づく状態推定

3.1　はじめに

本章では，ロボットが獲得した情報から，状態空間 $\mathcal{X}$ 内での現在の状態を推定するためのアルゴリズムを設計，実装する．ただしここでは，自己位置 (x, y, θ) についての推定のみを扱う．本章の構成を以下に示す．

3.2 節では，4 足ロボットリーグの環境が，自己位置推定に対してどのような性質を有するか説明する．

3.3 節では，自己位置推定によく用いられる数学モデルであるベイズ推定が，従来どのように自己位置推定アルゴリズムとして実装されていたかを説明し，これらがどのような誤差要因に対して有効であったか，そうでなかったかを考察する．

3.4 節では，考察に基づき，実際に自己位置推定のための数学モデルを設計する．

3.5 節では，設計した数学モデルに基づき，実際に自己位置推定アルゴリズムを設計，実装する．

3.6 節では，自己位置推定に用いる画像処理を設計，実装する．

3.7 節では，本章のまとめを行う．

3.2　4 足ロボットリーグにおける自己位置推定問題

ここでは，前章の問題設定に加えて，議論に必要な 4 足ロボットリーグの性質を説明する．

3.2.1　得られる情報の性質

もし色が完全に抽出できて，さらに画素が非常に細かく，ロボット座標系とカメラの関係が完全に分かるという前提を置くと，ランドマークについては，ランドマークの色ラベル画像上での幅と位置から，Fig.3.1 のように，ロボット座標系でのランドマークの距離と方向が決まる．また，ゴール板については，Fig.3.2 のように，ゴールの右端と左端の色ラベル画像上での方向から，ロボット座標系でのゴール板の位置と向きが一意に決まる．したがって，ゴール板からはフィールド座標系でのロボットの位置を一意に決定することができる．

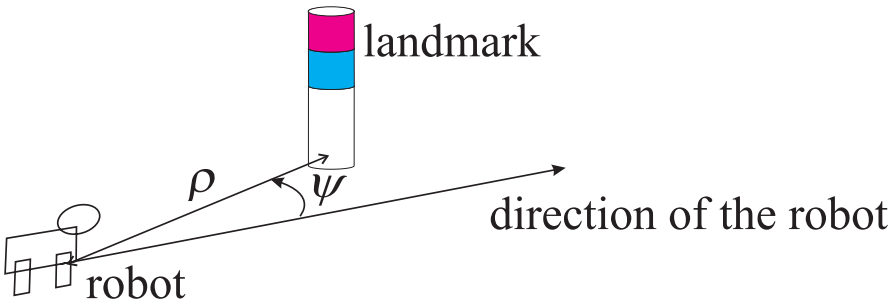

Fig. 3.1　ランドマークから得られる情報

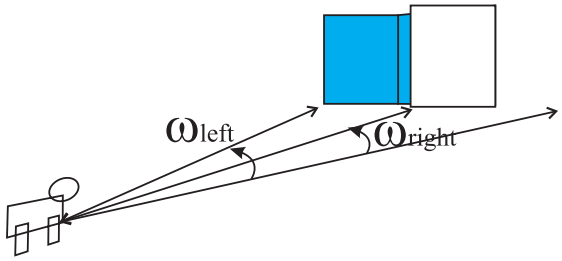

Fig. 3.2　ゴールから得られる情報

正確に位置を求めるだけならば，ロボットは静止して，カメラの揺れが止まるのを待って，二つのランドマークの位置計測を行って三角測量をすればよい．し

かし，ゲーム中のどのタイミングでも，このような観測を行う時間の余裕はない．つまり，ロボットは，ほとんどの情報を歩行中に得なければならない．また，カメラの視野が狭いので，ゴール板が視野にすべて入る場合は限られている．その他，あらゆる原因から計測誤差（次節で例を挙げる）が発生するため，理想的な場合には一意にロボットの位置が決定できるような情報源からでも，実際には決定は不可能である．

3.2.2　ロボットの移動に関する性質

ロボットの歩行では，同じ行動要素を選択しても移動量にばらつきが生じるが，このばらつきはわずかなものである．むしろ，試合中のロボット同士の接触を，ロボットが直接知覚できないことが問題となる．接触によって，ロボットの位置が変化することがあるが，このような直接知覚できない位置変化は，接触前後の情報の矛盾から，間接的に求めるしかない．

また，試合中には，反則等を行ったロボットを人間が強制移動するという場合がある．この場合も，ロボットは直接知覚できない．この問題は，kidnapped robot problem[Engelson 92, Fox 00] と呼ばれ，これを解決できると，ロボットは環境中で全く位置が分からない状態から，自己位置推定が行えるようになる．

強制移動と接触とは，ロボットの変位の大きさが異なるが，ロボットは両者を区別できないため，同じアルゴリズムで，大きな位置のずれと，小さい位置のずれに対応しなければならない．しかし，これは困難な問題である．強制移動に対応するには，矛盾が生じた時点で，過去に推定した自己位置を棄却しなければならず，接触に対応するなら過去の推定を利用しなければ無駄が大きい．したがって，強制移動と接触の両方に対応するには，同じアルゴリズムで相反する特性を達成しなければならない．

3.2.3　問題に対するアプローチ

以上をまとめると，4 足ロボットリーグにおける自己位置推定問題は，次の性質を有する問題と言える．

(1) 環境内の計測対象の位置，形状は，既知．

(2) ひとつの計測対象からは，自己位置情報が部分的に決定できる．しかし，一般に測定誤差は大きい．

(3) 移動しながら位置推定を行う必要がある．

(4) 知覚，予想不可能な様々な大きさの移動誤差が生じる．

このうち，(1),(2),(3) については，ベイズの定理で確率的に扱うことができる．ベイズの定理を用いた自己位置同定は，マルコフ自己位置推定（Markov localization）と呼ばれ，ロボカップのみならず，他の移動ロボットにおいても多くの実装例がある．本研究で提案するアルゴリズムも，マルコフ自己位置推定で対応できる部分は，なるべくマルコフ自己位置推定を用いて設計することとする．また，(4) については，マルコフ自己位置推定とは別の枠組みで対応することとする．

3.3　マルコフ自己位置推定

ロボットの位置 (x, y)，方向 θ で構成される空間 $\mathcal{X}_{\rm robot}$ を定義したとき，マルコフ自己位置推定では，点 $\forall \ell \in \mathcal{X}_{\rm robot}$ において，その点におけるロボットの存在確率密度 $bel(\ell)$ が求められる．ロボットが観測を行ったり，移動するごとに $bel(\ell)$ は更新され，ロボットは自己位置の情報が必要なときに，最新の $bel(\ell)$ を利用する．

3.3.1　更新則

観測によりロボットが，何らかの情報源から情報を得て，その値が η であるとき，ベイズの定理 [野田 92] により，$bel(\ell)$ は，

$$bel(\ell) \longleftarrow bel(\ell|\eta) = \frac{bel(\ell)p(\eta|\ell)}{\int_{\mathcal{X}_{\rm robot}} bel(\ell')p(\eta|\ell')d\ell'} \tag{3.3.1}$$

のように更新される．ここで $p(\eta|\ell)$ は，状態 ℓ において値 η を得る確率密度である．この式の分母は，正規化する役割を果たす．ベイズ統計学では，$p(\eta|\ell)$ は設計者によって事前に，主観的に与えられる（主観説の立場，subjective view）．実装された $p(\eta|\ell)$ はロボットの主観となり，ロボットはそれを事前知識として，センサ情報の値 η を上式で自己位置の情報に変換することになる．$p(\eta|\ell)$ を今後，観測確率モデルと呼ぶ．

また，ロボットが移動すると，$bel(\ell)$ の分布形状も，それによって移動する．このときの $bel(\ell)$ の更新式は，

$$bel(\ell) \longleftarrow \frac{\int_{\mathcal{X}_{\rm robot}} bel(\ell')p(\ell|\ell', a)d\ell'}{\iint_{\mathcal{X}_{\rm robot}} bel(\ell'')p(\ell'|\ell'', a)d\ell''d\ell'} \tag{3.3.2}$$

で表される．ここで，$p(\ell|\ell', a)$ は，ℓ' において行動 $a \in \mathcal{A}$ を行った後, ロボットの状態が ℓ である確率密度である．$p(\ell|\ell', a)$ も $p(\eta|\ell)$ 同様，設計者によって与えられる確率密度分布である．$p(\ell|\ell', a)$ を今後，移動確率モデルと呼ぶ．

マルコフ自己位置推定の出力は，確率密度分布 $bel(\ell)$ そのものか，あるいは領域 $X \subset \mathcal{X}_{\rm robot}$ 内にロボットの位置，方向 $\ell_{\rm robot}$ が存在する確率 $Bel(X)$ である．$Bel(X)$ は，

$$Bel(X) = \int_X bel(\ell)d\ell \tag{3.3.3}$$

で計算できる．

3.3.2 確率モデルの設計に関する議論

観測確率モデル $p(\eta|\ell)$ は，設計者が主観的に与えると先に述べたが，これは，設計者が，η に含まれる誤差をどれだけ考慮するのかで，設計に自由度があるということである．実環境では様々な原因から，なにかを計測したときの値に誤差が生じる．たとえば，4 足ロボットリーグの環境で，ロボットが画像中でランドマークの幅を求めて，ランドマークとの距離を計測しようとするときの誤差要因は，筆者が考えられるだけでもかなり存在する．ここでは，誤差要因の一部を，従来手法の扱い方の違いで分類して，従来手法がどのような誤差要因に対応し，対応しなかったかを説明する．

比較的小さな誤差　以下のような誤差要因が，この範疇に入る．

- ロボットの頭部の動きが速く，画像がぶれて画像中での物体の位置や大きさが変化する．
- ランドマークと背景の境界部分の画素値が，ランドマークと背景の色が混ざって平均化され，色抽出が行えない．
- 画像の量子化誤差，標本化誤差．

従来のマルコフ自己位置推定あるいはカルマンフィルタ [加藤 87] による自己位置推定では，主にこれらの要因による誤差について扱われてきた．これらの誤差は，ある一定の範囲で，正規分布に従って発生するものが多いため，容易に観測確率モデルが作成できる．

ただし，この観測確率モデルは，非常に短時間に連続して得られる情報に対しても成り立っていなければならない．しかし，カメラから得た連続する画像に対し，同じ画像処理をしてある値を得たとき，これらの値は互いに似ている場合がほとんどであり，観測確率モデルの予測する値のばらつきに従わない．ERS-210 のカメラは，32[ms] 毎に画像処理が可能である．この程度の速さであると，一度ランドマークなどがカメラに映ると，しばらく同じランドマークの情報が連続的に得られる．これらの情報には，同様の誤差要因のために，同じ傾向の誤差が含まれていると考えられる．例えば，ぶれた画像の直後に得られる画像は，やはりぶれている可能性が高い．このような同じ傾向の誤差が生じた情報に対して，連続して式 (3.3.1) を適用すると，$p(\eta|\ell)$ に極大値がある場合に，ロボットの位置とは別の箇所に確率密度の非常に大きな点が発生する．このような $bel(\ell)$ は，曖昧

さを表現しているとは言い難い．

この現象を避けるために，式 (3.3.1) を一度適応した後のしばらくは，得られる情報が，観測確率モデル $p(\eta|\ell)$ に従うことが明らかになるまで式 (3.3.1) を再び計算しないなどの対策がとられる．しかしこのためには，いつになれば情報がモデルに従う生起確率で得られるかという基準を設定しなければならない．この設定次第では，重要な情報が含まれる画像が捨てられたり，逆に確率密度の収束が発生したりする．また，ロボットが静止しているときに，その場所で二つ以上の情報を用いないという方法もあるが，これも重要な情報が含まれる画像が捨てられる恐れがある．

比較的大きな誤差　これは，以下のような原因で生ずる誤差を指す．

- 他機がランドマークを隠す（オクルージョン）．
- 画像の端にランドマークの一部分だけが映る．

これらの誤差要因は，しばしばまったく正しくない計測値を発生させる．しかも，オクルージョンなど，自身とは異なる移動体によって発生するものは，その生起確率をモデル化できない．従来法の多くは，これらを見つけ次第，その情報を無視するか，あるいはあまり起こらない事象として，観測確率モデルで考慮せず，これらの原因によって自己位置に大きな誤差が生じた後，いかに早く誤りから回復するかという枠組みで扱われている（例えば [Lenser 00]）．しかし，4 足ロボットリーグやオフィスのように，他の移動体が多い環境では，これらの方法では対応できない．また，たとえ計測対象が一部しか観測できない状況でも，あるときはこれが非常に重要な情報となることがある．これらが推定に利用できないどころか，これらによって悪影響が出るというのは非常に問題である．

定常誤差　4 足ロボットリーグでは，照明条件が変化し，色抽出が完全にできず，ランドマークの一部が欠けて見えるということが起こる．その日の天気や，昼夜の違いにより，ある計測値の平均値が長期的に変化する．つまり，ある日統計をとったランドマークと画像中のランドマークの幅の関係が，次の日に統計をとると，異なっているということが起こりうる．このような誤差に対しては，事前のキャリブレーションによって極力取り除くという方法が，4 足ロボットリーグでよく行われる．しかし，キャリブレーションの手間は，極力少ないほうがよいし，自律ロボットとしての価値を考えると，人間が関与することを少なくする必要があるのは，序論で述べたとおりである．

観点を変えると，小さな誤差のところで述べた連続して同じ要因による誤差が生じるという問題も，数秒間の定常誤差とも言える．また，大きな誤差要因についても，全く同様のことが起こる．結局，カメラの情報からベイズ推定を行う場合，定常誤差は，あらゆるレベルで問題となるのである．

3.4　一様分布観測確率モデルに基づく自己位置推定

前節の議論から，様々な誤差要因に対応する自己位置推定法を設計するには，

- 互いに独立でない情報が連続して入力されることに対応する
- 普段から起こりうることに悪影響を受けない

ことが必要である．これに対して，観測確率モデルを一様分布で表現することで，対応することを提案する．なぜ一様分布がよいかという議論は，アルゴリズムの説明とともに述べていく．

3.4.1　更新則 I（マルコフ自己位置推定）

まず，観測確率モデル $p(\eta|\ell)$ が，

$$p(\eta|\ell) = \begin{cases} \dfrac{1}{\int_{L(\eta)} d\ell} & (\text{if } \ell \in L(\eta)) \\ 0 & (\text{else}) \end{cases} \tag{3.4.1}$$

という確率分布に従うとする．ここで $L(\eta)$ は，情報を得て，その値が η である可能性が否定できない領域である．例えば Fig.3.3 のように，人[*1]がランドマークを見て距離を η と知覚したとき，η と知覚しうる範囲が幅の広い狭いにかかわらず存在するので，これを $L(\eta)$ とする．また，上式が示すように，$L(\eta)$ 内での $p(\eta|\ell)$ の値は一定とする．

$p(\eta|\ell)$ が上式に従うとき，式 (3.3.1) の右辺は，

$$\frac{bel(\ell)p(\eta|\ell)}{\int_{\mathcal{X}_{\text{robot}}} bel(\ell')p(\eta|\ell')d\ell'} = \begin{cases} \dfrac{bel(\ell)}{\int_{L(\eta)} bel(\ell')d\ell'} & (\text{if } \ell \in L(\eta)) \\ 0 & (\text{else}) \end{cases} \tag{3.4.2}$$

となり，更新後の $bel(\ell)$ は，式 (3.3.1) から，

$$bel(\ell) \longleftarrow \begin{cases} \dfrac{bel(\ell)}{\int_{L(\eta)} bel(\ell')d\ell'} & (\text{if } \ell \in L(\eta)) \\ 0 & (\text{else}) \end{cases} \tag{3.4.3}$$

[*1] ここでロボットを例に出さないのは，ロボットがこのランドマークの距離計測ができるというのは，自明でないからである．

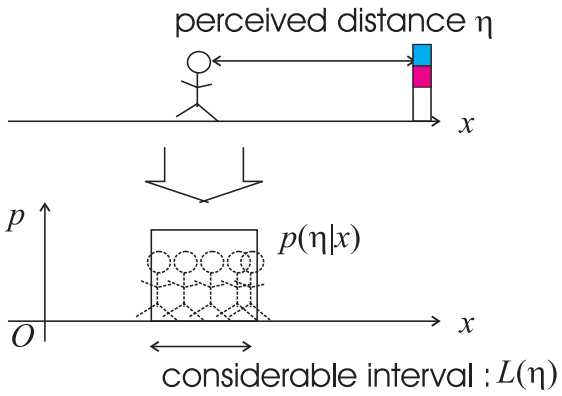

Fig. 3.3　観測確率モデルの例（一次元の例）

となる．この式を一次元状態空間で表現すると，Fig.3.4 のようになる．センサ値 η が入力されると，事前に推定されていた $bel(\ell)$ の各点 ℓ について,「センサ値 η を ℓ で得る」という事象がありえない部分の確率密度が 0 になる．そして，確率が全体で 1 になるように正規化される．この式では，同じ計測結果が続けて得られたとき，$bel(\ell)$ に変化は起きない．また，定常誤差が観測確率モデルに正しく反映されていれば，観測確率モデルと矛盾する情報が入らない限り，$bel(\ell) \neq 0$ である領域から真の位置が出るということもない．

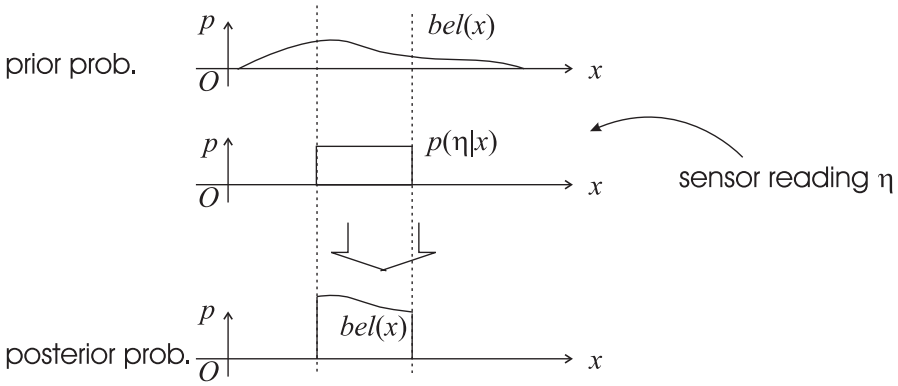

Fig. 3.4　観測確率モデルと自己位置の事前確率との作用（式 (3.4.3)）

3.4.2　更新則 II（マルコフ自己位置推定で対応できない部分）

ロボットが直接知覚できない位置変化は，位置変化前の推定位置と，変化後に得る情報の矛盾で間接的に知るしかないと以前に述べた．この「矛盾」は，更新

則 I で $bel(\ell) = 0$ となる領域が生じるため，

$$\forall \ell \in L(\eta) \textbf{ に対して } bel(\ell) = 0 \textbf{ となること}$$

であると，明確に定義できる．マルコフ自己位置推定では，本来このような事象が起こりえないようにモデル化するのであるが，ロボットが直接知覚できない位置変化がある以上，このようなことは現実に起こりうる．

矛盾が生じたときは，式 (3.4.3) に代わる別の更新則を適用する必要がある．あらゆる大きさの位置変化に対応するためには，以下の条件を同時に満たすことが望ましい．

- 位置変化が起こった後，なるべく短時間で，変化後の位置の近傍において，$bel(\ell) \neq 0$ となること．
- 位置変化が起こった後の $bel(\ell)$ が，なるべく曖昧にならないこと．

これらの条件を満たすために，$\forall \ell \in L(\eta)$ に対して $bel(\ell) = 0$ となるとき，更新前の $bel(\ell) \neq 0$ の範囲を x, y, θ 軸に沿って α 倍に拡大するという方法をとる．倍率 α については，ある決められた時間以内に $bel(\ell) \neq 0$ がロボットの位置，方向まで達するように決める．この方法だと，矛盾が続く場合，短時間のうちに分布がロボットの移動後の位置まで拡散する．一方，矛盾が一，二回で終われば，もとの $bel(\ell)$ の情報の損失は，それほど大きなものにはならない．

具体的には，どの場所にロボットが移動されても，確実にその位置まで分布を拡散するために，以下のような手続きを行う．まず，更新前に $bel(\ell) \neq 0$ である部分を x, y, θ 軸のそれぞれに投影したときの区間 $[x_{\min}, x_{\max}]$，$[y_{\min}, y_{\max}]$，$[\theta_{\min}, \theta_{\max}]$ を求める．そして，それぞれの区間の中心の座標 $(x_{\rm c}, y_{\rm c}, \theta_{\rm c})$ を持つ点 $\ell_{\rm c}$ と，区間の幅 $x_{\rm wid}, y_{\rm wid}, \theta_{\rm wid}$ を求め，

$$bel(\ell) = \begin{cases} \dfrac{1}{\int_{B(\eta)} d\ell} & (\text{if } \ell \in B(\eta)) \\ 0 & (\text{else}) \end{cases} \tag{3.4.4}$$

とする．ここで，領域 B は，$B = \{\ell(x, y, \theta) \mid |x - x_{\rm c}|\alpha/2 \leq x_{\rm wid}, |y - y_{\rm c}|\alpha/2 \leq y_{\rm wid}, |\theta - \theta_{\rm c}|\alpha/2 \leq \theta_{\rm wid}\}$ とする．つまり，Fig.3.5(A) のように，更新前の $bel(\ell)$ から，それが収まる直方体（各辺が x, y, θ 軸のいずれかに平行）を求めて，Fig.3.5(B)（この場合は $\alpha = 2$）のように，その直方体をその場で各辺について α 倍して，それを $bel(\ell) \neq 0$ の領域とする．

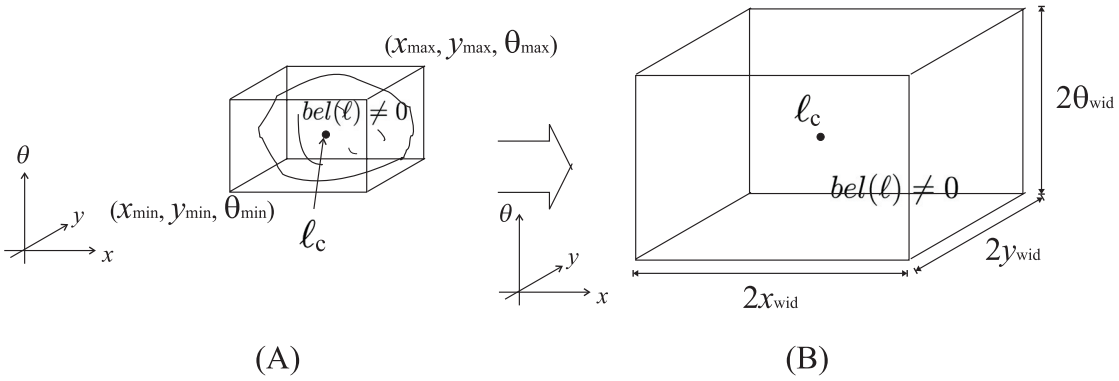

Fig. 3.5　矛盾の発生前 (A) と発生後 (B) の $bel(\ell)$

3.5 Uniform Monte Carlo Localization

モンテカルロ法で前節の更新則を実現する．まず，サンプルξ_i $(i = 1, 2, \ldots, N_{\rm sample})$を定義する．サンプルとは，確率論に関するモンテカルロ法では空間中に配置する点を意味する．モンテカルロ法は，サンプルになんらかのパラメータを持たせることで，空間中の関数を近似する．

ここでは，各サンプルは，確率$1/N_{\rm sample}$を受け持つこととする．同じ確率を受け持つサンプルを多数，空間中に配置し，Fig.3.6 のように，ある領域Xに存在するサンプルの数によって，その領域内にロボットの真の位置，向き$\ell_{\rm robot}$が存在する確率$Bel(X)$を表現する．つまり，サンプルξ_iの位置をℓ_iで表現すると，

$$Bel(X) = \frac{1}{N_{\rm sample}} \sum_{i=1}^{N_{\rm sample}} \delta[\ell_i, X] \qquad \left(\delta[\ell_i, X] = \begin{cases} 1 \text{ if } \ell_i \in X \\ 0 \text{ if } \ell_i \notin X \end{cases} \right) \tag{3.5.1}$$

とする．この式は，式 (3.3.3) の近似となっている．サンプルは，確率を持つので，厳密には点ではなく，空間的な拡がりをもつが，この拡がりの形状や，その内部の確率密度分布については，考慮しない．このモンテカルロ法の適用方法は，従来の Monte Carlo Localization とは異なるため，区別のため，Uniform Monte Carlo Localization（Unform MCL）と呼ぶ．

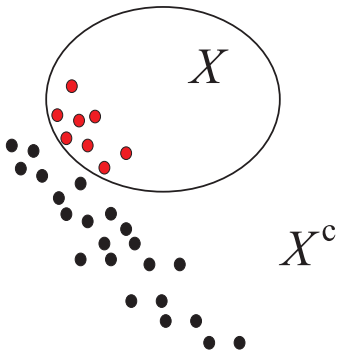

Fig. 3.6 ロボットの存在確率の計算方法

ここで，サンプル数$N_{\rm sample}$は，定数ではなく変数である．つまり，サンプルは消えたり新たに発生したりする．また，実時間で計算できるサンプル数の上限$N_{\rm max}$があると仮定する．また，最初サンプルは，$xy\theta$空間にランダムに配置され，$bel(\ell)$がこの空間中で一様であることを表現し，以後説明するサンプル操作によっ

て，$bel(\ell)$ を更新していく．

ベイズ推定の式 (3.4.3) は，全サンプル ξ_i $(i = 1, 2, \ldots, N_{\text{sample}})$ に対して次のように操作することで実装できる．

$$\text{if } \ell_i \notin L(\eta) \longrightarrow \text{erase}[\xi_i] \tag{3.5.2}$$

ここで，erase$[\xi_i]$ は，サンプル ξ_i を消去する操作を表す．つまり，サンプルの位置と入力されたセンサ値を比較して,「位置 ℓ_i にロボットが存在したと仮定して，センサ値 η を得る可能性」が否定できる場合，この位置にあるサンプル ξ_i を，Fig.3.7 で $\times$ 印のついたサンプルのように，消去する．また，サンプル数 N_{sample} は，サンプルを 1 つ消去するごとに 1 引くことによって管理される．

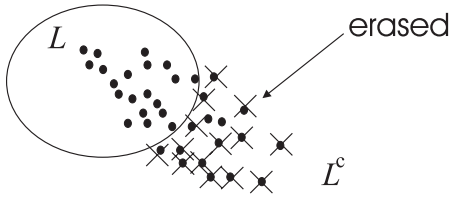

Fig. 3.7　センサ値入力時のサンプル操作

また，移動確率モデルを適用する式 (3.3.2) は，次のように実装される．まず，移動前のサンプルを n 分割する．ただし，$nN_{\text{sample}} \leq N_{\max}$ である必要がある．そして，新たなサンプル ξ_i $(i = 1, 2, \ldots, N_{\text{sample}})$ に対して，以下の操作を行う．

$$\ell_i \longleftarrow \text{choose}[\ell, p(\ell|\ell_i, a)] \tag{3.5.3}$$

ここで，choose$[\ell, p(\ell|\ell_i, a)]$ は，確率密度分布 $p(\ell|\ell_i, a)$ にしたがって，ℓ をひとつ選びだす操作を示す．$p(\ell|\ell_i, a)$ については，事前にロボットを移動させて，移動量のばらつきをとることで実装を行う．Fig.3.8 にその様子を示す．この図では，1 個のサンプルを 3 個に分裂させて，それぞれ $p(\ell|\ell_i, a)$ に従ってサンプルを移動させている．ただし，本来サンプルは $xy\theta$ 空間に属するため，移動前のサンプルの θ_i 値に応じて，サンプルはこの図よりも複雑な挙動を示す．

Uniform MCL では，$N_{\text{sample}} = 0$ になったときに，情報の矛盾が起こったとみなせる．このときの式 (3.4.4) に関する操作は，更新後の $bel(\ell) \neq 0$ とする領域内に，$N_{\text{sample}}^{\top}$ 個のサンプルをランダムに配置する．更新前の区間 $[x_{\min}, x_{\max}]$，$[y_{\min}, y_{\max}]$，$[\theta_{\min}, \theta_{\max}]$ は，全サンプルの位置について，サンプルが操作されるたびに x, y, θ それぞれの最大値，最小値を求めておくことで得られる．

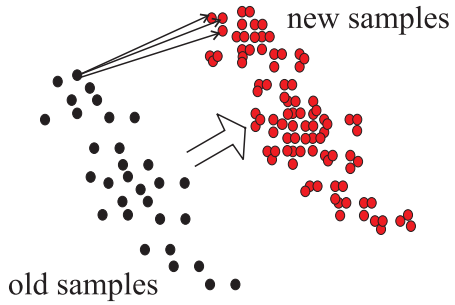

Fig. 3.8 サンプルの分裂と移動

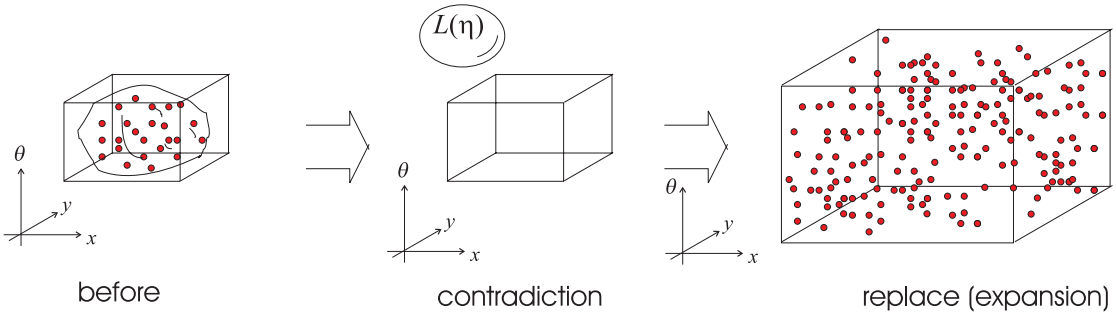

Fig. 3.9 サンプルが 0 個になったときの操作

3.6 画像処理アルゴリズムの設計

ここでは，ERS-210 のカメラ画像から，Uniform MCL に入力する情報を作成するアルゴリズムを設計する．本研究の目的である,「環境変化」への対応は，照明条件が変化しても，出力が Uniform MCL の使用する一様分布観測確率モデルと矛盾しないように，画像処理アルゴリズムを作成することで可能となる．通常であれば，これはあらゆる雑音を考慮して，画像処理に様々な工夫を行うことで対応するのであるが，この方法では，無数にある誤差要因の全てに対応できる保証がない．

Uniform MCL で用いる画像処理やセンサ処理では，そのような誤差要因をしらみつぶしにするよりも，むしろ，ある「前提条件」を設定して，それが成り立っている限り観測確率モデルと矛盾しない情報を画像から抽出するというアプローチをとる．この方法も，必ず全ての誤差要因に対応できないということには変わりはないが，処理のアルゴリズムは，非常に簡潔にできる．ここでは，実際に Uniform MCL で使用する画像処理アルゴリズムを簡単に説明して，その例を示す．

3.6.1 ランドマーク計測アルゴリズム

ランドマーク計測で求められる「前提条件」は，以下のようなものである．

- **ランドマーク周辺に，ランドマークの色のいずれかに誤って色抽出が行われるような物体が存在しないこと．**

これは，ランドマークの周囲に全く同色のものがなければ，色抽出の設定を，ランドマークに関係する色を抽出しにくいようにすることで，ほぼ完全に成り立たせることができる．この仮定から，画像処理の設計を開始する．

まず，Fig.3.10 右に見られる白い長方形のように，ピンク色の一番大きな領域を覆う長方形の位置を求める．この長方形の中心を G と表現する．G を画像座標系からロボット座標系 Σ_{robot} へ変換すると，ロボット座標系でのランドマークの方向に関する情報が得られる．ところが，カメラ座標系とロボット座標系の位置が異なるため，Fig.3.11 に示すように，画像中の点 G の位置が同じであっても，ランドマークのカメラからの距離によって，ロボット座標系でのランドマークの方向に曖昧さが残る．ランドマークまでの距離が求まればこの問題は解消するが，Fig.3.10 のように抽出すべき部分の欠けた色抽出画像からは，正確に求めることができな

い.

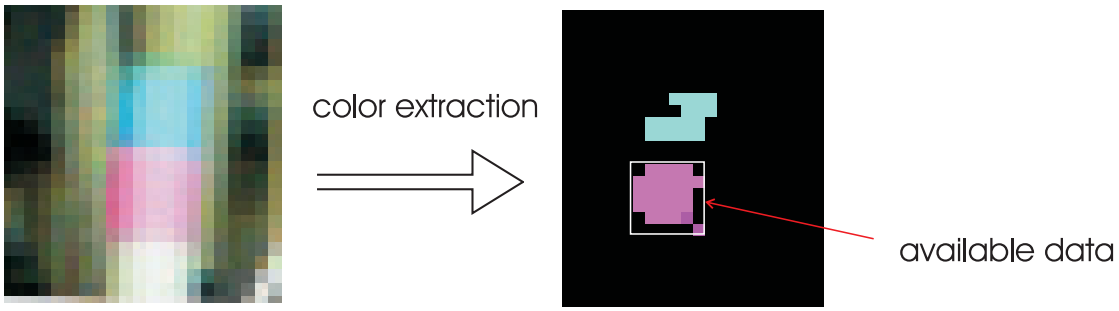

Fig. 3.10 ランドマークの色抽出画像

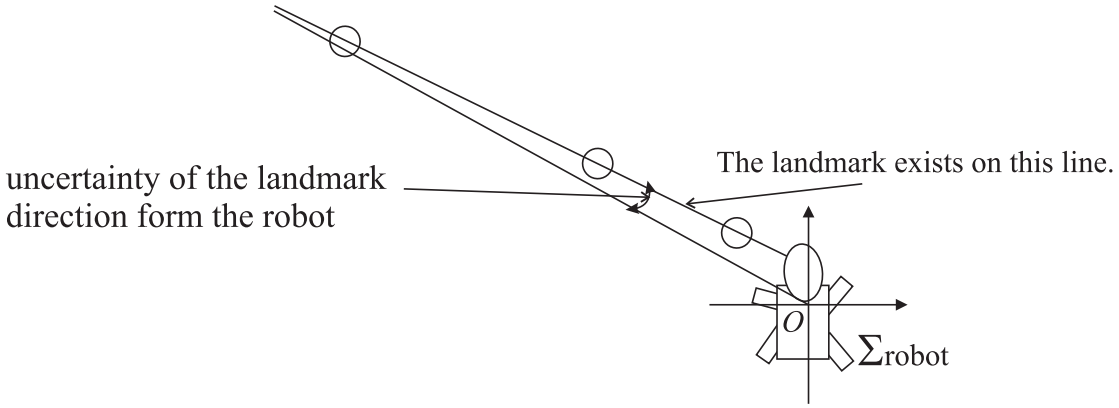

Fig. 3.11 視差に関する誤差

そこで，方向は，ランドマークが無限遠点にあるときと，観測できる最も近い距離（300[mm]）にあるときを仮定して計算し，ロボット座標系での方向の差を観測確率モデルに反映させることで対処する．また，点 G が，ランドマークのピンク色の部分の中心に存在することは保証できないので，中心からのずれに対する誤差も，観測確率モデルに反映させる．これは，ランドマークがカメラから 300[mm] の距離にあることを仮定し，ランドマークの任意の場所に G が存在することを想定して，Fig.3.12 のように最大誤差を求める．この誤差を計算すると ± 9.9[deg] となるので，この誤差も観測確率モデルに反映させる．その他，歩行中にロボットの胴体が揺れて，ランドマークの方向の観測に数度誤差が生じることも考え，考慮しなければならない誤差 e_ψ は，20[deg] 程度となる．

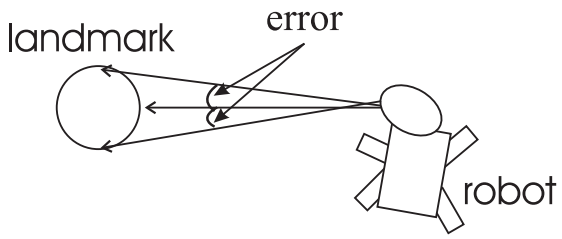

Fig. 3.12　点 G の位置に関する誤差

次に，観測されているランドマークを識別する．カメラ座標系 $\Sigma_{\rm cam}$ と，ロボット座標系 $\Sigma_{\rm robot}$ の関係から，画像中で，ランドマークがどれだけ傾いているかを求める．この求め方は，[上田 01] に詳しい記述がある．点 G を通り，求めたランドマークの傾きに平行な直線（Fig.3.13 に示す）上の，他の色のピクセル数を数える．最もピクセル数の多い色と，その色とピンク色の上下関係から，観測しているランドマークを識別する．

これで，どのランドマークを観測して，それがどの方向 ψ にあるか，曖昧さを含みながらも求まったことになる．これを式 (3.5.2) の手続きによって，全サンプル ξ_i $(i = 1, 2, \ldots, N_{\rm sample})$ に対し，次のように反映させる．

(1) サンプル ξ_i の位置 ℓ_i にロボットが存在すると仮定して，ランドマークの方向 ψ_{ξ_i} を求める．

(2) $|\psi_{\xi_i} - \psi| > e_\psi \longrightarrow {\rm erase}[\xi_i]$

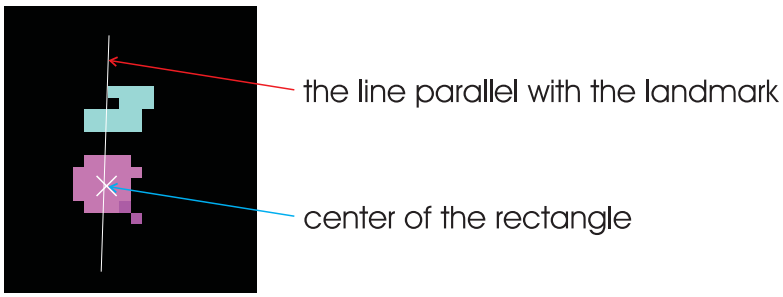

Fig. 3.13　画像中でランドマークと平行な直線

最後に，ランドマークとの距離に関する情報を求めるが，前述のように，これを正確に求めることは非常に困難である．色抽出が完全であれば，Fig.3.14 のように，色のついた長方形領域であれば，どの辺や対角線，あるいは面積を画像中で

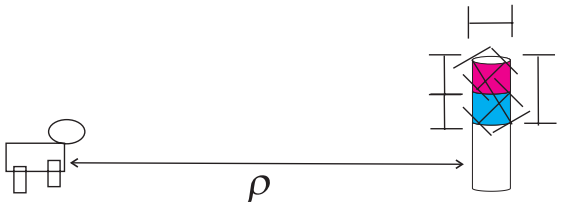

Fig. 3.14 ランドマークの距離計測に使用できる情報

計測しても，距離を求めることができる．しかし，色抽出が不完全だと，これらのどれについても，実際計測されるべき長さで計測される保証がない．また，遠方のランドマークになるほど，一つの画素の色抽出失敗が，計測に大きな支障をきたす．

しかし，前述の前提条件から，長方形領域の対角線は，実際計測されるべきよりも小さな値で出力されると言える．一方，長方形領域の辺の長さは，カメラが傾くと実際より長く計測される可能性があるので，その保証はない．そこで，ピンク色領域[*2]の対角線の長さを計測して，これを自己位置推定に反映させることを考える．

もしロボットの位置が既知であれば，理想的な色抽出画像でのピンク色領域の長さ $w_{\max}$ が決定できる．Fig.3.10 のピンク色の部分について，ある線分（必ずしも対角線でなくてよい）の長さを計測したとき w であったとする．このとき，必ず $w \leq w_{\max}$ となる．そこで，式 (3.5.2) の手続きを，全サンプル ξ_i $(i = 1, 2, \ldots, N_{\text{sample}})$ に対して，以下のように行うことができる．

(1) サンプル ξ_i の位置 ℓ_i に，ロボットが存在すると仮定して，最大幅 $w_{\max}$ を求める．

(2) $w > w_{\max} \longrightarrow \text{erase}[\xi_i]$

この手続きは，ランドマークの距離測定の誤差を考慮して，誤差範囲外のサンプルを消去するというよりも，あるサンプルの位置と計測値が前提条件と矛盾している場合に，サンプルを消去するという意味合いが強くなっている．

ランドマーク計測によって，サンプルが減少する様子を，Fig.3.15 に示す．ここではサンプルの上限は 10,000 点とした．Fig.3.15 の上図は，長い矢印の位置，向

[*2] ここでピンク色としたのは，一般にどの照明条件でも抽出しやすいからである．

きのロボットが,「+」印の位置にあるランドマークを一度観測して得られるサンプルの分布である．雑音の考慮から，ランドマークから近いサンプルが残っている．

中図は，上図の状態から別のランドマークを観測したときのサンプルの分布である．二つのランドマークから，ロボットの向きはほとんど決定されている．様々な要因を考慮して推定を行っても，複数のランドマークを計測することで，自身の位置を大まかに推定できることが分かる．

下図は，同じランドマークを観測したときに，雑音の影響で中図のときの計測値とは別の値が得られた後のサンプルの分布である．この図のように，同じランドマークを計測しても，誤差が発生して計測値がずれた場合，もしそれが考慮されている範囲内の誤差であれば，両方の計測値のどちらかと矛盾するサンプルは消滅するため，結果として推定が進む．

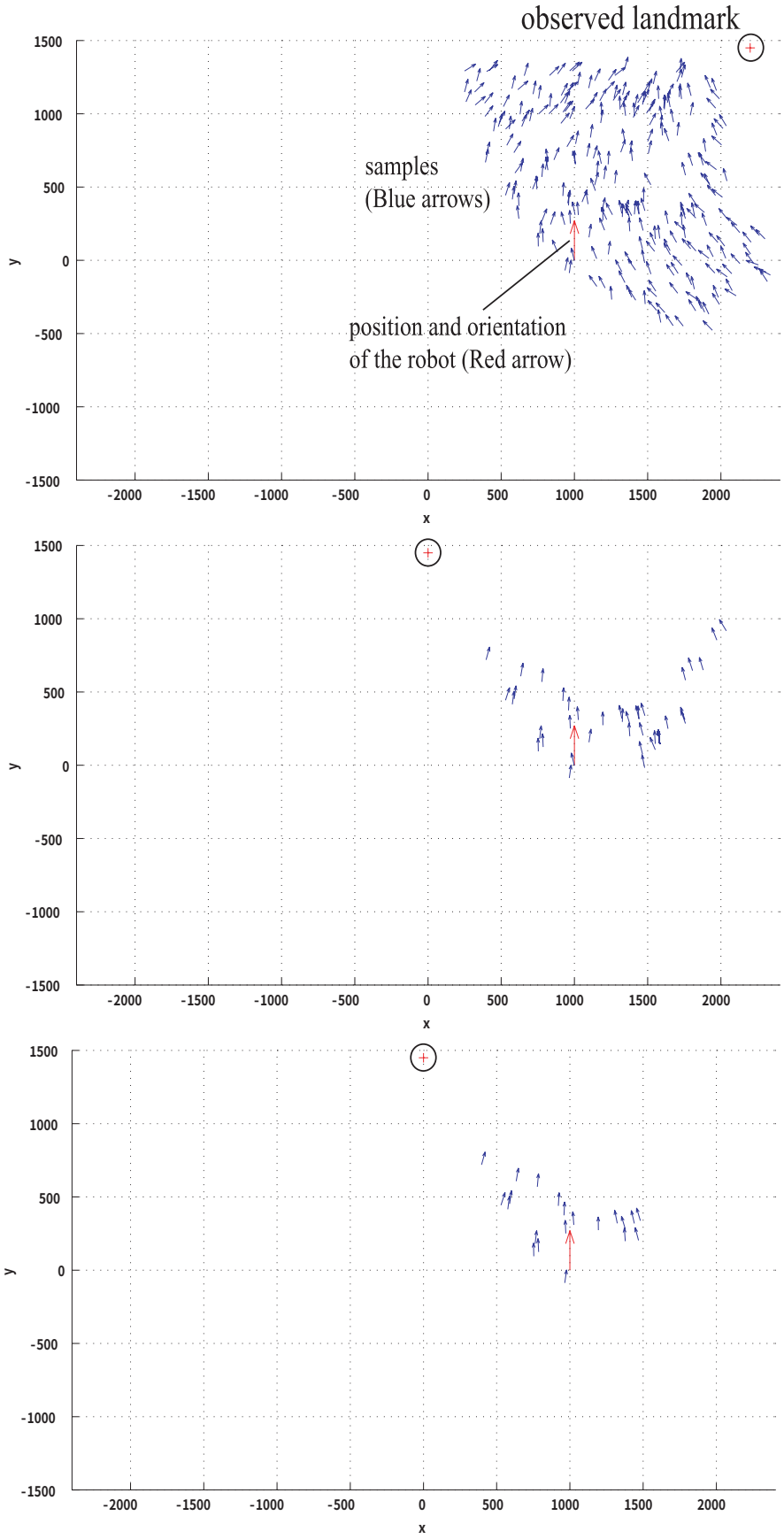

Fig. 3.15 サンプルの消滅する様子

3.6.2 ゴール位置計測アルゴリズム

ゴールの計測の場合も，ランドマークのものと同じ前提条件が用いられる．まず，色抽出画像から，ゴールを示す空色あるいは黄色の一番大きな領域について，画像の右端と左端の画素の位置を求める．その二つ画素の画像上の位置から，ランドマークの方向を求めたときと同じ方法で，それぞれのロボットからの方向$\omega_{\rm right}, \omega_{\rm left}$を計算する．色の領域がゴールを完全に抽出しているわけではないので，ゴール周辺にゴールと同色の物体がないと仮定すると，本来のゴールの右端は，$\omega_{\rm right}$ より右側，本来のゴールの左端は，$\omega_{\rm left}$ より左側にあるはずである．ただし，ランドマークの方向計測と同様，誤差が生じるため，正確には，Fig.3.16 のように，ゴールの両端に許容誤差を足した方向の範囲内に，$\omega_{\rm right}, \omega_{\rm left}$ の値が存在するはずである．

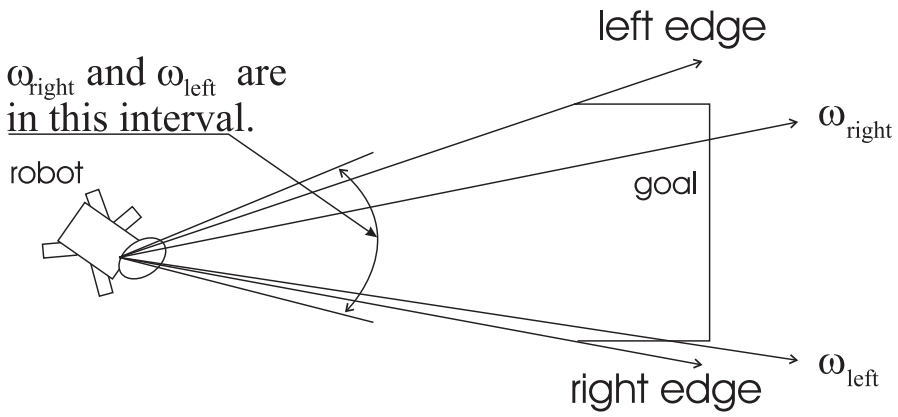

Fig. 3.16 計測された方向の許容範囲

この許容誤差を，ランドマーク同様 e_ω で表し，20[deg] とする．このとき，どちらかのゴールについて，方向 $\omega_{\rm left}, \omega_{\rm right}$ が観測によって得られたとき，式 (3.5.2) の手続きを，サンプル ξ_i $(i = 1, 2, \ldots, N_{\rm sample})$ に対して，以下のように行う．

(1) サンプル ξ_i の位置 ℓ_i から，ゴールの方向 $\omega_{{\rm left}\xi_i}, \omega_{{\rm right}\xi_i}$ を求める．

(2) $\omega_{{\rm left}\xi_i} < \omega_{\rm left}$ or $\omega_{{\rm right}\xi_i} > \omega_{\rm right} \longrightarrow {\rm erase}[\xi_i]$

また，この他に，ゴールの色の領域の幅から，ランドマーク同様距離を求めるアルゴリズムもあるが，これは，ランドマークの方法とほとんど同じであるので，説明を省略する．

3.7　おわりに

本章では，以下のことを行った．

- 4 足ロボットリーグにおける自己位置推定問題の性質の説明
- マルコフ自己位置推定の説明
- 物体の位置計測に伴う誤差要因に関する議論
- Uniform Monte Carlo Localization（Uniform MCL）の設計
- Uniform MCL に用いるための画像処理アルゴリズムの設計
- 設計したアルゴリズムの実装

Uniform Monte Carlo Localization と，ある仮定を置いた上で信頼できる画像処理を組み合わせることで，4 足ロボットリーグのフィールドでロバスト性が期待できる自己位置推定を実装した．Uniform MCL は，従来扱われていた誤差要因よりも，取り扱いが難しいものについて扱えることや，連続して同じ要因による誤差が混入したセンサ値を入力した場合の悪影響がないという特性を持つ．これは，カルマンフィルタでは構造的に不可能である．また，他のマルチモーダルの手法に対しては，より簡潔なアルゴリズムで上記の性質が得られることが，Uniform MCL の優れている点である．この簡潔さによって，高速に演算ができることが期待できる．

実装したアルゴリズムの評価については，第 6 章のシミュレーション，第 7 章の実機実験で行う．また，本章で設計したアルゴリズムは，第 5 章で用いられる．

第4章　動的計画法による行動計画

4.1　はじめに

本章では，現在の状態 $\boldsymbol{x} \in \mathcal{X}$ が既知であるという前提で，その状態においてロボットがとるべき行動（方策 π）を，動的計画法によって設計する．動的計画法によって状態既知を前提とする方策を求める手法は，従来から行われているものである．しかし，曖昧さの考慮というテーマにおいては，方策 π は「どのような状態推定方法にも依存しない，そのタスクに対する普遍的な知識」という新たな意味を持つ．動的計画法は，ある条件のもとに解の最適性が保障されるため，この「普遍的な知識」を得るためには適した手法であると言える．

本章は，以下のように構成される．

4.2 節では，動的計画法について説明を行う．

4.3 節では，フォワードロボットのボールへの接近と，キーパー行動についての二つの具体的なタスクについて，方策を得る．これらの方策は，次章以降で用いられる．

4.4 節では，本章のまとめを行う．

4.2 動的計画法

4.2.1 最適化問題とベルマン方程式

ここでは，動的計画法 [Bellman 57] が適用される問題と，動的計画法の目的について述べる．まず，状態 $\forall \boldsymbol{x} = (x_1, x_2, \ldots, x_m) \in \mathcal{X}$，出力ベクトル $\boldsymbol{u} = (u_1, u_2, \ldots, u_n) \in \mathcal{U}$ に対して，連続時間での状態方程式が

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) \tag{4.2.1}$$

で与えられているとする．この式は，ある時間 t での状態 $\boldsymbol{x}(t)$ が与えられると，その将来の状況は，過去の履歴には無関係であるとできるので，マルコフ過程である．$\boldsymbol{x}(t), \boldsymbol{u}(t)$ に対して，関数 $R(\boldsymbol{x}(t), \boldsymbol{u}(t))$ を定義し，汎関数

$$V = \int_0^T R(\boldsymbol{x}(t), \boldsymbol{u}(t))dt \tag{4.2.2}$$

を，最大[*1]にするための出力 $\boldsymbol{u}(t)$ を，各瞬間全てにおいて求めることを考える．ここで，T は固定とする．この問題は最適化問題と呼ばれ，時間 T までの全過程が終了しないと評価が決まらないのに，各瞬間に出力 $\boldsymbol{u}(t)$ を決定する必要があるという性質を有する．

求める出力が求まったとして，それを $\boldsymbol{u}^*(t)$ と表現すると，それに付随して状態の最適経路 $\boldsymbol{x}^*(t)$ も得られる．$0 \leq t_1 < t_2 \leq T$ を満たす時間 t_1, t_2 を考える．このとき，最適経路中の点 $\boldsymbol{x}^*(x_1), \boldsymbol{x}^*(x_2)$ の間で，$[t_1, t_2]$ において $V = \int_{t_1}^{t_2} R(\boldsymbol{x}(t), \boldsymbol{u}(t))dt$ を最大にする最適経路は，$\boldsymbol{x}^*(t)$ に一致する．つまり，$[0, T]$ の間のどの部分区間も，評価関数を最大にする最適経路は，$\boldsymbol{x}^*(t)$ 上に存在する．これは，最適性の原理 I と呼ばれる [近藤 84].

最適性の原理 I が，終端の点 $\boldsymbol{x}(T)$ が固定された上で成り立ち，ある時点 t で将来の状況が過去の履歴に無関係であることを注意すると，t における最適な出力 $\boldsymbol{u}^*(t)$ は，その時点での状態 $\boldsymbol{x}(t)$ のみに依存する．これは，最適性の原理 II と呼ばれる．

最適経路 $\boldsymbol{x}^*(t)$ が見つかっていると仮定すると，区間 (t, T) における V の最大値

[*1]関数 R の定義次第であるが，文献によっては最小という記述になっている．ここでは R は後に述べる報酬と関係があるため，最大とした．

V^* は，最適経路 $\boldsymbol{x}^*(t)$ で決定できるので，

$$\begin{aligned} V^*(t, \boldsymbol{x}^*(t)) &= \max \int_t^T R(\boldsymbol{x}(t), \boldsymbol{u}(t))dt \\ &= \int_t^T R(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t))dt \end{aligned} \tag{4.2.3}$$

と表現できる．

一方，微小時間 Δt をとり，$\boldsymbol{x}(t+\Delta t)$ を考えると，テイラー展開により

$$\begin{aligned} \boldsymbol{x}(t+\Delta t) &= \boldsymbol{x}(t) + (d\boldsymbol{x}/dt)\Delta t + o(\Delta t) \\ &= \boldsymbol{x}(t) + \boldsymbol{g}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t))\Delta t + o(\Delta t) \quad (\text{式 (4.2.1) より}) \end{aligned} \tag{4.2.4}$$

とできる．ここで，$o(\Delta t)$ は高次の項である．また，$V^*(t+\Delta t, \boldsymbol{x}^*(t+\Delta t))$ は，

$$V^*(t+\Delta t, \boldsymbol{x}^*(t+\Delta t)) = V^*(t, \boldsymbol{x}^*(t)) + \{\tilde{\nabla} V^* \cdot \boldsymbol{g}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t))\}\Delta t + o(\Delta t) \tag{4.2.5}$$

となる．ここで，$\tilde{\nabla} V^* = (\partial V^*/\partial x_1, \partial V^*/\partial x_2, \ldots, \partial V^*/\partial x_m)$ である．

また，式 (4.2.3) から，

$$\begin{aligned} V^*(t, \boldsymbol{x}^*(t)) &= \max \int_t^{t+\Delta t} R(\boldsymbol{x}(t), \boldsymbol{u}(t))dt + \max \int_{t+\Delta t}^T R(\boldsymbol{x}(t), \boldsymbol{u}(t))dt \\ &= \max R(\boldsymbol{x}(t), \boldsymbol{u}(t))\Delta t + V^*(t+\Delta t, \boldsymbol{x}^*(t+\Delta t)) \end{aligned} \tag{4.2.6}$$

が成り立つ．

この式に，式 (4.2.5) を代入して，Δt で割ると，

$$0 = \max R(\boldsymbol{x}(t), \boldsymbol{u}(t)) + \tilde{\nabla} V^* \cdot \boldsymbol{g}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t)) + o(\Delta t)/\Delta t \tag{4.2.7}$$

となり，$\Delta t \to 0$ とすると，

$$0 = \max R(\boldsymbol{x}(t), \boldsymbol{u}(t)) + \tilde{\nabla} V^* \cdot \boldsymbol{g}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t)) \tag{4.2.8}$$

が得られる．したがって，$\boldsymbol{u}(t)$ が最適な出力であるためには，条件式

$$\max R(\boldsymbol{x}(t), \boldsymbol{u}(t)) + \tilde{\nabla} V^* \cdot \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t)) = 0 \tag{4.2.9}$$

を満たす必要がある．この式は，ベルマン方程式と呼ばれ，$\boldsymbol{u}(t)$ が最適であれば，状態 $\boldsymbol{x}(t)$ において $\boldsymbol{u}(t)$ を出力することに対して与えられる評価と，$\boldsymbol{u}(t)$ を出力す

ることで微小移動した後の V^* の減少量が釣り合うことを意味している．動的計画法は，この方程式を満たす出力 $\boldsymbol{u}^*(t)$ を求めることを目的とした手法群であり，具体的なアルゴリズムを指す言葉ではないが，終端状態（$t = T$ のときの状態）が決まっており，どのような手法を用いても，終端状態から時間を逆転する向きに $\boldsymbol{u}^*(t)$ が求まっていくことは共通している．

また，この手法群に関係する方法として，強化学習が盛んに研究されている [Sutton 98]．これは，出力 $\boldsymbol{u}(t)$ をある方法に従って選択して，陰に設定された評価関数 R の出力を観測することで，試行錯誤的に最適な出力 $\boldsymbol{u}^*(t)$ を求める方法と解釈できる．強化学習では，R に相当するものは，状態 $\boldsymbol{x}$ で出力 $\boldsymbol{u}$ を選択したときの即時報酬と呼ばれる．また，V は状態価値と呼ばれ，これは式 (4.2.3) で分かるように，その後得られる報酬の合計のことを意味する．人工知能に関しては，強化学習が多く取り上げられ，（R や V などの記号も含めて）このような用語の用いられ方をするので，強化学習は扱わないものの，本論文でもそれに習うこととする．

4.2.2 具体的な解法（価値反復）

動的計画法は，問題設定によって様々な解法がある．第 2 章の問題設定では，出力（行動）a が離散的であるとしたので，離散的な出力用に，ベルマン方程式を書き直す必要がある．また，現在のところ，状態空間を離散化しない手法で，最適な出力を得る保証のあるものがないことが [Boyan 95] で述べられているため，ここでは確実に計画問題を解くため，状態空間 $\mathcal{X}$ を有限個の状態 s_i $(i = 1, 2, \ldots, N)$ から構成される離散状態空間 $\mathcal{S}$ に離散化する．また，各行動に要する時間が一定のため，時間も等間隔に離散化する．

このとき，状態方程式 (4.2.1) に相当するものは，現在の状態 s から，行動 a を選択したとき，次の状態 s' に遷移するという状態遷移のモデルである．s' は一つに決定的に決まる必要はなく，別の状態に遷移する可能性があってもよい．ここで，次のどの状態に，どれだけの確率で遷移するかが既知であれば，状態遷移モデルは，状態 s から行動 a を行ったときに s' に遷移する確率

$$\mathcal{P}^a_{ss'} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \Re \tag{4.2.10}$$

$\mathcal{P}^a_{ss'}$ の集合で与えられる．また，報酬については，現在の状態 s から，行動 a を選択したとき，次の状態 s' に遷移することに対して与えられ，

$$\mathcal{R}^a_{ss'} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \Re \tag{4.2.11}$$

で定義される．また，求めるものは，ある状態で，どの行動を選択するのが最適かということで，これは，最適方策 $\pi^*(s) \in \mathcal{A}$ と表現する．

このとき，ベルマン方程式は，

$$V^*(s) = \max_a \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + V^*(s')] \tag{4.2.12}$$

となる．この式の a は，最適方策によって選択されているとする．これも式 (4.2.9) と同様に，方策が最適であれば，状態遷移前後の状態価値の差の期待値と，即時報酬の期待値が釣り合うことを意味している．また，このとき，最適方策は，

$$\pi^*(s) = \arg_a \max \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + V^*(s')] \tag{4.2.13}$$

と表現できる．

この形式のベルマン方程式に対し，最も単純かつ多用されるものは，価値反復である．これは，ある状態の価値を求めるために，その状態から，ある行動をした後の状態価値と報酬の和の期待値を求めて，その期待値が最大のものを状態価値とするという手続きを，終端状態を除くすべての状態に対して，収束まで繰り返すものである．ひとつの状態に対する一回の計算を式にすると，

$$V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + V^*(s')] \tag{4.2.14}$$

となる．価値反復のアルゴリズムを，Fig.4.1 に示す．このアルゴリズムは，収束する保証があるものである．

V を，任意の値で初期化．例えば，$\forall s \in \mathcal{S}^+$ に対して $V(s) = 0$.

繰り返し：

$\Delta \leftarrow 0$

各 $s \in \mathcal{S}$ について：

$v \leftarrow V(s)$

$V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + V^*(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

$\Delta < \theta$（正の小さな数）ならば，繰り返しを終了

Fig. 4.1 価値反復の擬似コード（[Sutton 98] より抜粋）

4.3　サッカーのための行動計画

ここでは，動的計画法を用いて，

- フォワードが，シュートするゴールの方向を向いてボールを確保する
- キーパーが，ゴールを守る

という二種類のタスクに対して行動計画を行う方法を，実装レベルで具体的に説明する．説明中の価値や報酬の単位は，ロボットの歩数である．ここでは，確実に状態価値関数と方策を得るため，前述のように収束が保証されている価値反復を用いることとする．

4.3.1　フォワードのボールへの接近行動

ロボットがボールをカメラで観測し続けたまま，最少歩数でボールに近づくというタスクに対して，計画を行う．ロボットは，ボールに近づくとき，フィールドの周囲の壁にぶつからないように行動する必要があり，また，ボールに接近したときに相手側のゴールの方向を向いていなければならない．

状態空間の離散化

まず，状態空間 $\mathcal{X}$ 中で，フォワードが行動する領域を Table 4.1 のように設定する．$x < -1750[\mathrm{mm}]$ への領域は，ルール上，フォワードが侵入してはならないこ

Table 4.1　各状態変数の定義域

状態変数	定義域	単位
x	$[-1750, 2100)$	[mm]
y	$[-1350, 1350)$	[mm]
θ	$[-180, 180)$	[deg]
r	$[120, 2400)$	[mm]
φ	$[-75, 75)$	[deg]

とになっている[*2]．また，$|\varphi|$ を 75[deg] 以下とした理由は，ボールトラッキングが

[*2]厳密には，「キーパ以外のロボットは，味方ゴールエリア内で，足を 3 本以上ついてはならない」というルールであるので，少しの進入は許される．

それ以上の角度では不安定になるからである．

上記の定義域の離散化を Table 4.2 のように行う．この離散化方法で，状態数は，全部で 801,900 個になる．以後，各軸に関する各離散区間を，例えば x 軸であれば，x_i $(i = 1, 2, \ldots, 22)$ のように，添え字をつけて表現する．

Table 4.2　各状態変数の離散化

状態変数	間隔	区間の数
x	175[mm]	22
y	180[mm]	15
θ	20[deg]	18
r	変則的（Table 4.3 参照）	9
φ	10[deg]	15

Table 4.3　r の離散化

index	区間 [mm]
1	[120,200)
2	[200,250)
3	[250,300)
4	[300,400)
5	[400,600)
6	[600,900)
7	[900,1300)
8	[1300,1800)
9	[1800,2400)

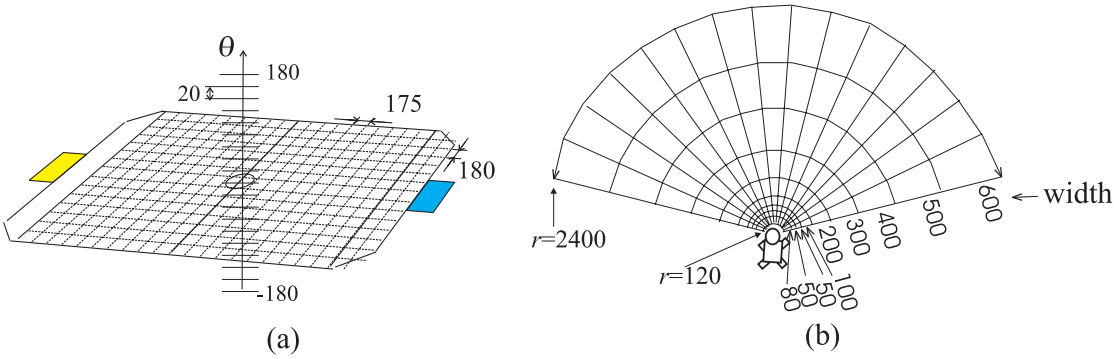

Fig. 4.2　状態空間の離散化

終端状態の定義

ある離散状態について，各状態変数に関する区間の中心値で構成される状態ベクトルを $(x_{\text{center}}, y_{\text{center}}, \theta_{\text{center}}, r_{\text{center}}, \varphi_{\text{center}})$ と表す．この状態ベクトルに対して，以下の条件がすべて成り立つとき，その状態を終端状態とする．

- ボールの位置に関して
  - $|\varphi_{\mathrm{center}}| < 45[\mathrm{deg}]$
  - $r_{\mathrm{center}} < 200[\mathrm{mm}]$
- 自己位置に関して
  - ロボット座標系 Σ_{robot} での相手ゴール（フィールド座標系 Σ_{field} において $(2100, 0)$ ）の方向が，$\pm 30[\mathrm{deg}]$ 以内

ただし，ロボットが壁際にいるときは，自己位置に関する条件を緩和して，

- $y_{\mathrm{center}} < -1170[\mathrm{mm}]$ のときに，θ_{center} が区間 $(-60, 30)[\mathrm{deg}]$ に含まれる
- $y_{\mathrm{center}} > 1170[\mathrm{mm}]$ のときに，θ_{center} が区間 $(-30, 60)[\mathrm{deg}]$ に含まれる
- $x_{\mathrm{center}} > 1925[\mathrm{mm}]$ かつ $y_{\mathrm{center}} \leq 0[\mathrm{mm}]$ のときに，θ_{center} が区間 $(30, 90)[\mathrm{deg}]$ に含まれる
- $x_{\mathrm{center}} > 1925[\mathrm{mm}]$ かつ $y_{\mathrm{center}} > 0[\mathrm{mm}]$ のときに，θ_{center} が区間 $(-90, 30)[\mathrm{deg}]$ に含まれる

という条件であっても終端状態とする．ここで，r の区間 $[120, 200)[\mathrm{mm}]$ については，終端状態以外のとき以外は，入ってはいけないこととする．つまり，これらの状態は，定義域から除外する．

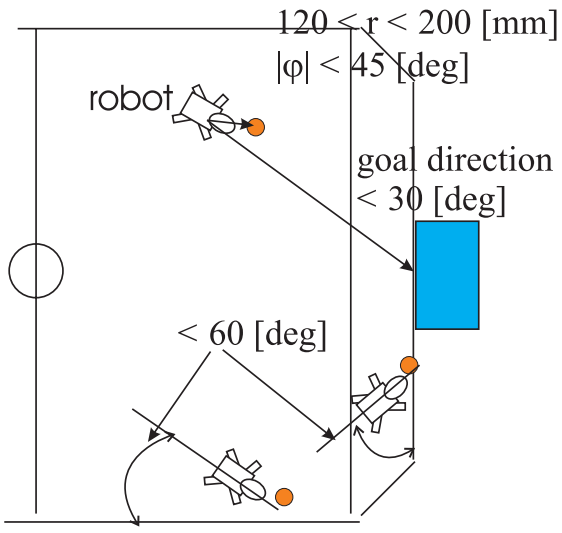

Fig. 4.3　終端状態

状態遷移確率の計算

ロボットの移動，ボールの相対的な位置変化は，すべて誤差がないという仮定をおいて方策を求める．この方法では，連続状態空間中の状態遷移はすべて決定論的になる．しかし，離散状態空間だと，ある離散状態 s 内に現在の状態ベクトル $\boldsymbol{x}$ があるとき，s 内のどの状態が $\boldsymbol{x}$ であるかということは確定できないため，状態遷移も確率的になる．

次のような手続きで，状態遷移確率 $\mathcal{P}^a_{ss'}$ を求める．

(1) $\forall s'$ に対して，カウンタ $c_{s'}$ を設け，0 に初期化

(2) 以下を n 回繰り返し

　(a) 状態ベクトル $\boldsymbol{x} \in s$ をランダムに選択

　(b) 点 $\boldsymbol{x}$ を行動 a によって状態ベクトル $\boldsymbol{x}'$ に遷移

　(c) $\boldsymbol{x}' \ni s' \Rightarrow c_{s'}{+}{+}$

(3) $\mathcal{P}^a_{ss'} = c_{s'}/n$

Fig. 4.4　状態遷移確率の計算方法

点を $\boldsymbol{x} = (x, y, \theta, r, \varphi)$ から状態ベクトル $\boldsymbol{x}' = (x', y', \theta', r', \varphi')$ へ遷移させる計算は，移動前のロボット座標系での行動 a による移動量（Table 2.3 参照）を $(\delta x, \delta y, \delta\theta)$ とすると，

$$\begin{pmatrix} x' \\ y' \\ \theta' \\ r' \\ \varphi' \end{pmatrix} = \begin{pmatrix} x + \delta x \cos\theta - \delta y \sin\theta \\ y + \delta x \sin\theta + \delta y \cos\theta \\ \theta + \delta\theta \\ \sqrt{(r\cos\varphi - x' + x)^2 + (r\sin\varphi - y' + y)^2} \\ \arctan[(r\sin\varphi - y' + y)/(r\cos\varphi - x' + x)] - \delta\theta \end{pmatrix} \tag{4.3.1}$$

と計算できる．

ここで，この計算は，壁際などのロボットが自由に歩行できない場所では適用できない．また，ロボットが複雑な形状をしているため，壁に衝突した後のロボットの位置，方向を計算することも困難である．このようなときには，以下のような対処方法がある．

- ロボットを壁に向かって歩行させて，衝突前後の位置，方向を記録して統計をとり，$\mathcal{P}^a_{ss'}$ を求める．
- 壁に衝突する行動を選択できなくする．

前者は，行動の種類が多くて大変な手間となるので，後者を選択し，壁に衝突する可能性のある行動を選択したときに負の無限大の報酬を与えることとする．このようにすると，ロボットやボールの移動量が，移動前の x, y の値に依存しなくなるため，$\mathcal{P}^a_{ss'}$ の計算を全離散状態に対して行う必要が無くなる．一方この方法には，必要なときにロボットが壁に近寄らなくなるという短所もある．

上記の選択に従い，Fig.4.4 の手続きを以下のように実装する．

- $\mathcal{P}^a_{ss'}$ を，自己位置 $\ell(x, y, \theta)$ に関する遷移確率 $\mathcal{P}^a_{\ell\ell'}$ と，ボールの位置 $b(r, \varphi)$ に関する遷移確率 $\mathcal{P}^a_{bb'}$ に分ける．
- 自己位置に関しては，区間 x_i と y_j を適当に選んで，θ_k $(k = 1, 2, \ldots, 18)$ と行動 a_m $(m = 1, 2, \ldots, 41)$ の全組み合わせに対して，Fig.4.4 の手続きを行う．x, y の他の区間については，区間 x_i と y_j に関する計算結果から，相対的に求めることができる．
- ボールの位置に関しては，r_i $(i = 1, 2, \ldots, 9), \varphi_j$ $(j = 1, 2, \ldots, 15), a_k$ $(k = 1, 2, \ldots, 41)$ の全組み合わせに対して，Fig.4.4 の手続きを行い，$\mathcal{P}^a_{bb'}$ を求める．

報酬の定義

報酬 $\mathcal{R}^a_{ss'}$ については，ロボットが一回歩行動作を行うたびに-1 を与える．また，以下のような状況を起こす可能性がある行動に対しては，負の無限大の報酬を与える．

- Table 4.1 で設定した定義域を出る（壁への衝突を含む）
- ボールがロボットに接触する（$r < 200$[mm] で，終端状態ではない場合）

計画結果

完全に収束するまで Mobile Pentium Ⅲ 1.2GHz を搭載した PC で 69 分 39 秒かかり，価値反復の回数は，176 回であった．Fig.4.5 に，収束したときの状態価値関数の一部を示す．この図は，ボールをフィールド座標系で $(0, -1200)$ の地点に置

いたとき，ロボットが各位置でボールの方向を向いているときの状態価値を計算したグラフである．$r < 120$[mm] や $r \geq 2400$[mm] となる部分は，状態価値を -50 としてプロットした．また，Fig.4.6 に，計画で得られた方策から得られるボールへのアプローチ行動の例を示す．

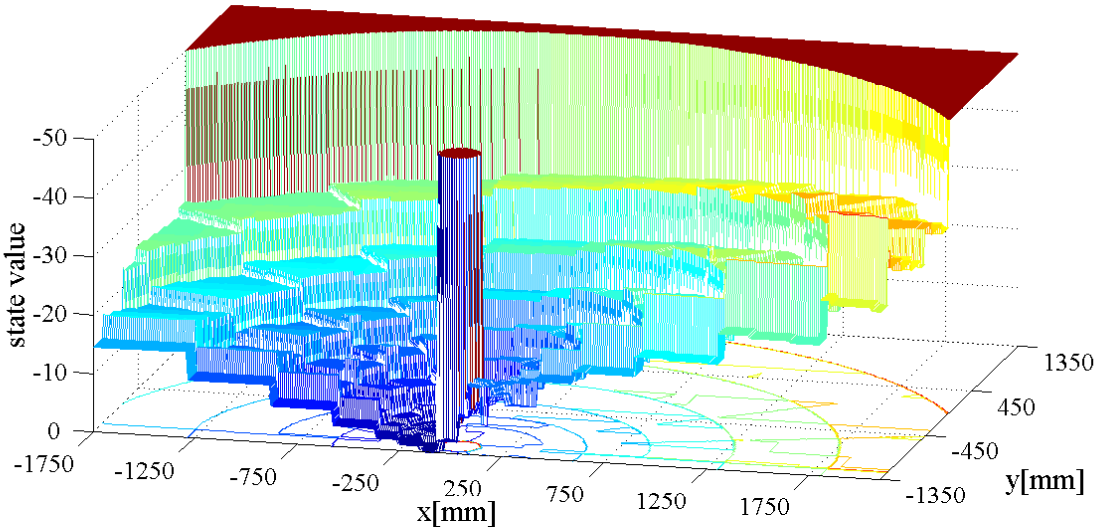

Fig. 4.5　状態価値関数（状態価値の軸の向きに注意）

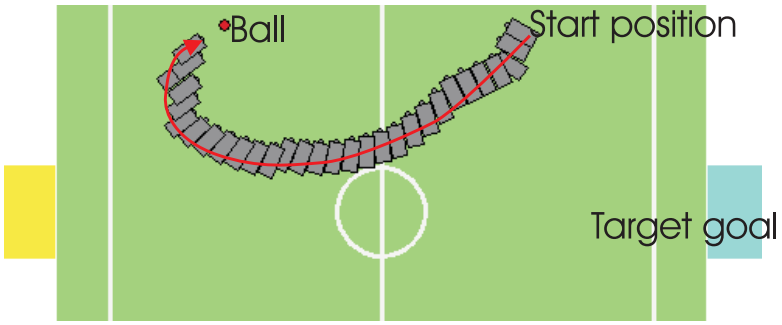

Fig. 4.6　計画によって得られたロボットの軌道

定義域外でのフォワードの行動

フォワードロボットは，Table 4.1 の定義域外でも行動決定しなければならない．そこで，ロボットが Table 4.1 で定めた定義域から外に出たときは，最も近い区間の方策を用いる．ただし，ボールの方向 φ の定義域からボールが出ている状態のときは，ロボットはその場で回転して，ボールの方向を修正することとする．

$r < 200$[mm] のときは，終端状態か，進入不可の状態なので方策が定まっていないため，終端状態ならばボールを運ぶために前進（WalkForward），他の場合はボールとの接触を避けるために後退（WalkBackward）というルールを追加する．また，ボールを見失ったときは，その場で回転して，ボールを探すこととする．

4.3.2　キーパー行動

キーパーの行動計画では，ゴールへの帰還とボールへのアプローチの二つの行動を同時に計画する．キーパーは，ゲーム中，以下のような高度な決定を行う必要がある．

- ボールがどれだけ近づいたらゴールから出てボールを確保するか
- ゴールから出た状態で，ボールを発見したときに，ボールに向かうべきか，ゴールに向かうべきか
- ボールが遠くにあるとき，ゴールのどの部分を自身の体で塞ぐべきか

このようなことをハンドコーディングする場合，設計者はロボットの行動決定全体を設計する必要があるが，動的計画法を用いると，報酬，終端状態とすべき状態と，終端状態の状態価値を設計すれば，そこに達するまでの行動決定は，価値反復によって求めることができる．

状態空間の離散化

まず，状態空間 $\mathcal{X}$ 中で，キーパーが行動する領域を Table 4.4 のように設定する．φ の範囲がフォワードよりも広いのは，キーパーのタスクがボールへの接近だけではないからである．また，キーパは遠いボールを追う必要性が少ないので，r の範囲は狭くなっている．

上記の定義域の離散化を Table 4.5 のように行う．ボールの位置が上記の区間外のときは，ボールが観測できないこととする．キーパーの場合はボールが観測できなくても，ゴールに帰還，待機するというタスクが存在するため，ボールが観測できない状態に対しても，x, y, θ に対して上記の離散化を行う．

この離散化方法で，ボールが観測可能なときの状態 2,973,348 個と，ボールが観測できないときの状態 6084 個の合計 2,979,432 個になる．ただし，$x > 2100$[mm],

Table 4.4 各状態変数の定義域

状態変数	定義域	単位
x	$[1000, 2400)$	[mm]
y	$[-1350, 1350)$	[mm]
θ	$[-180, 180)$	[deg]
r	$[120, 2020)$	[mm]
φ	$[-92, 92)$	[deg]

Table 4.5 各状態変数の離散化

状態変数	間隔	区間の数
x	100[mm]	14
y	100[mm]	27
θ	20[deg]	18
r	100[mm]	19
φ	8[deg]	23

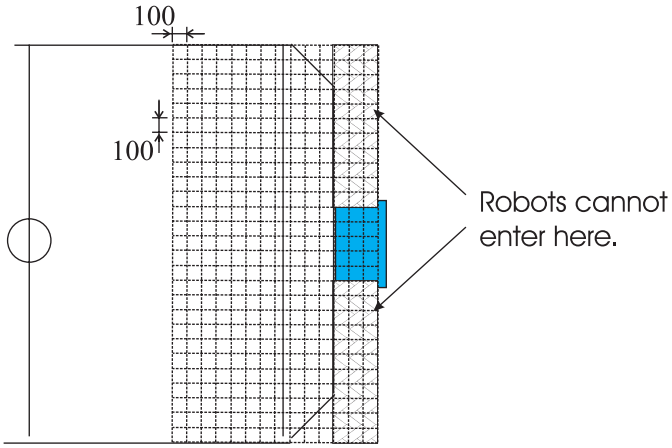

Fig. 4.7 ロボットが入れない領域

$|y|$ >300[mm] の部分にはロボットが入れず，この離散化方法には無駄があるが，ここではメモリの無駄よりもアルゴリズムを簡略化できる利点を優先する．

終端状態の定義

キーパーの終端状態として，Fig.4.8 に見られる 3 種類の状態を考える．

a. ボール確保（ボール確保状態）

b. ボールが発見できなくて，ゴールの中央に待機（ゴール待機状態）

c. ボールを遠方に発見して，ボールとゴールの間を塞ぐ（ゴール守備状態）

それぞれを，ある離散状態の中心の状態ベクトル $(x_{\mathrm{center}}, y_{\mathrm{center}}, \theta_{\mathrm{center}}, r_{\mathrm{center}}, \varphi_{\mathrm{center}})$ が，以下の条件を満たすときと定義する．

a. ボール確保状態のとき

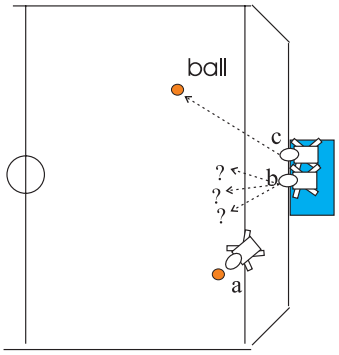

Fig. 4.8　キーパーの終端状態

- $|\varphi_{\text{center}}| < 45[\text{deg}]$
- $r_{\text{center}} < 200[\text{mm}]$
- $(x_{\text{center}}, y_{\text{center}}, \theta_{\text{center}})$ に対して，以下のいずれかが成り立つ
  - $|\theta_{\text{center}}| > 135[\text{deg}]$ のとき
  - ロボット座標系 Σ_{robot} での味方ゴール（フィールド座標系 Σ_{field} において $(2100, 0)$ ）の方向が，$\pm 135[\text{deg}]$ 以上

b. ゴール待機状態

- ボール観測不可能，$2100 < x_{\text{center}} < 2300[\text{mm}]$，$-100 < y_{\text{center}} < 100[\text{mm}]$，$|\theta_{\text{center}}| > 150[\text{deg}]$

c. ゴール守備状態

- $2100 < x_{\text{center}} < 2200[\text{mm}]$，$|\theta_{\text{center}}| > 150[\text{deg}]$
- 以下のいずれかが成り立つ
  - $\varphi_{\text{center}} < 12[\text{deg}]$
  - $\varphi_{\text{center}} < 0[\text{deg}]$ で $y_{\text{center}} > 50[\text{mm}]$
  - $\varphi_{\text{center}} \geq 0[\text{deg}]$ で $y_{\text{center}} < -50[\text{mm}]$

ここで，フォワードと同様に，r の区間 $[120, 200)[\text{mm}]$ については，終端状態以外のとき以外は，入ってはいけないこととする．つまり，これらの状態は，定義域から除外する．

これらの終端状態は，設計の都合から設けたもので，キーパー行動という，大きなタスクから見ると，これらの状態は終端状態ではないので，終端状態とされた状態間には価値の差異が生じるはずである．そこで，著者の経験から，歩数換算で次のように状態価値を与える．

a. ボール確保状態：0

b. ゴール待機状態：-15

c. ゴール守備状態：-10

ただし，この終端状態のうち，ゴール待機状態とゴール守備状態については，ボールの位置によっては，なにか行動をとったときに状態価値が上がる可能性があるので，式 (4.2.13) を適用して，最適行動を求める必要がある．つまり，これらの状態の一部は，状態価値を操作しないだけで，真の意味で終端状態ではない．

報酬の定義

報酬は以下のように与える．

- ボールを見失う: -5
- ボールもゴールも観測できない状態で一回行動（自殺点の恐れあり）: -5
- 上の条件以外のとき，一回行動: -1
- 壁にぶつかる．状態空間から出る．: -∞

計画結果

完全に収束するまで，前述の PC で，2 時間 1 分 12 秒を要した．このときの価値反復の回数は，80 回であった．状態価値関数の例を Fig.4.9 から Fig.4.13 までに示す．

Fig.4.9,4.10 は，ボールを $x = 1500$[mm]，$y = 500$[mm] の位置に置いたときの，ロボットの位置に対する状態価値を示したものである．ロボットの向きはそれぞれ，$\theta = -180$[deg]，$\theta = 0$[deg] に固定してある．また，ϕ, r が定義域内のときは，ボールが観測されているときの状態価値を表示している．ボールが観測できる領域とできない領域で不連続な部分が見られるなど，計算された状態価値関数が，複

雑な形をしていることが分かる．

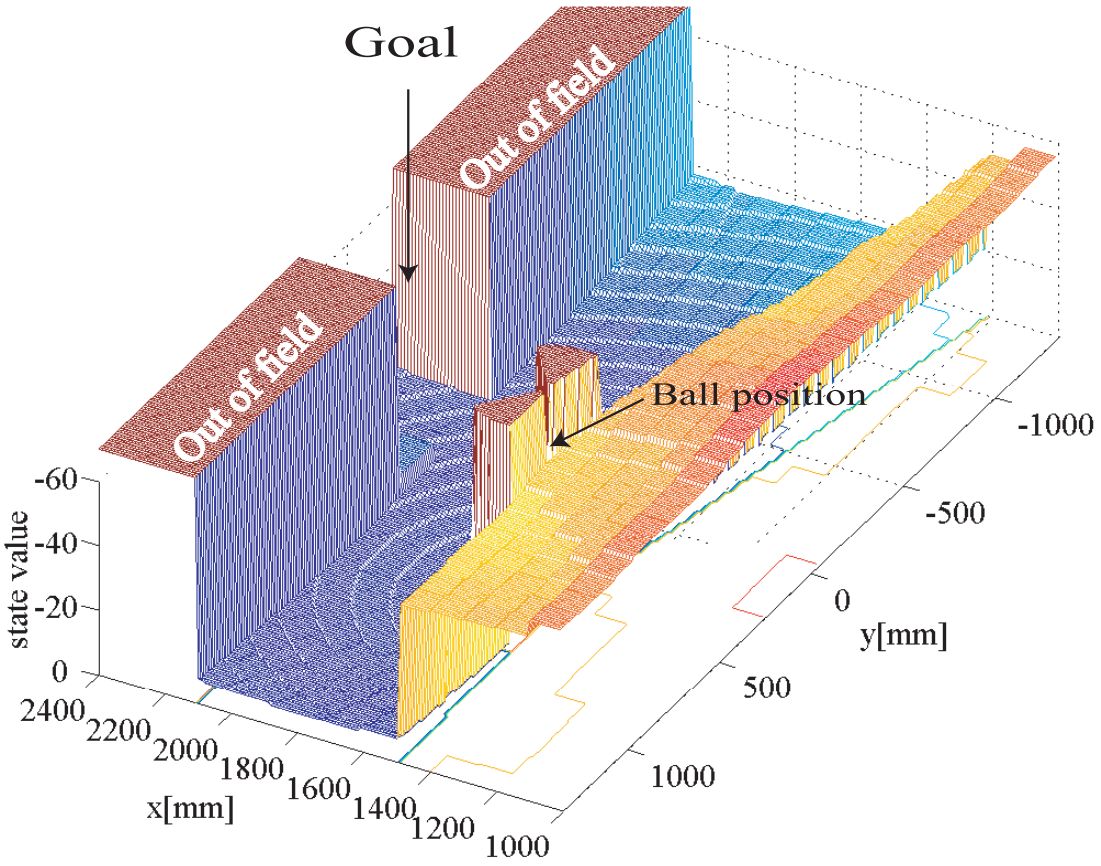

Fig. 4.9 $\theta = -180$[deg] の状態価値関数（状態価値の軸の向きに注意）

見方を変えると，Fig.4.11 や Fig.4.12 のようなグラフも得られる．Fig.4.11 は $\phi = 0$[deg] となる方向にロボットが向いているときのもので，フォワードと同様に，ボールに向かって価値が高くなっている．Fig.4.12 は，ボールがないときに，$\theta = -180$[deg] であるときのもので，ゴールに向かって価値が高くなっている．x の小さい部分では，ゴール前よりも両側の部分の価値が高いが，これは，自殺点の発生しやすさを報酬に考慮した結果が反映されたものである．

また，Fig.4.13 は，ボールがフィールド座標系で $x = 1100$[mm]，$y = 1300$[mm] の位置にあり，ロボットの向きが $\theta = -180$[deg] で位置 (x, y) に存在するときの状態価値関数である（この図では，定義域外の状態価値を仮に+10 として表示している）．ϕ, r が定義域内のときは，ボールが観測されているときの状態価値を表示している．このグラフからは，ゴール内とボール付近の二つに極値が存在してい

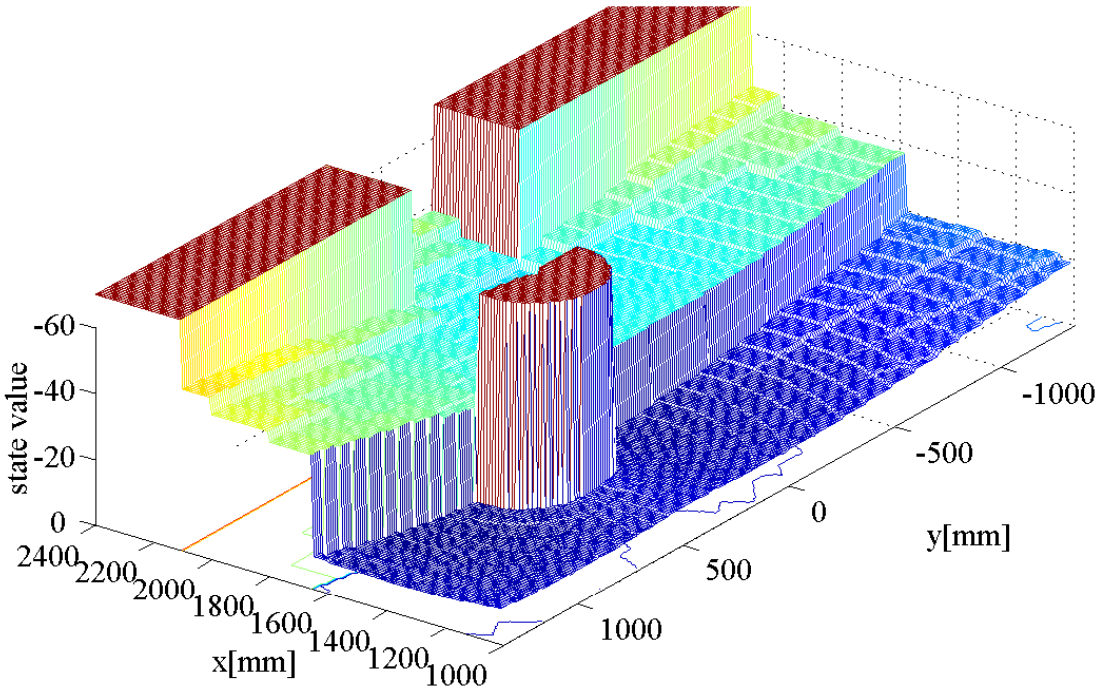

Fig. 4.10 $\theta = 0$[deg] の状態価値関数

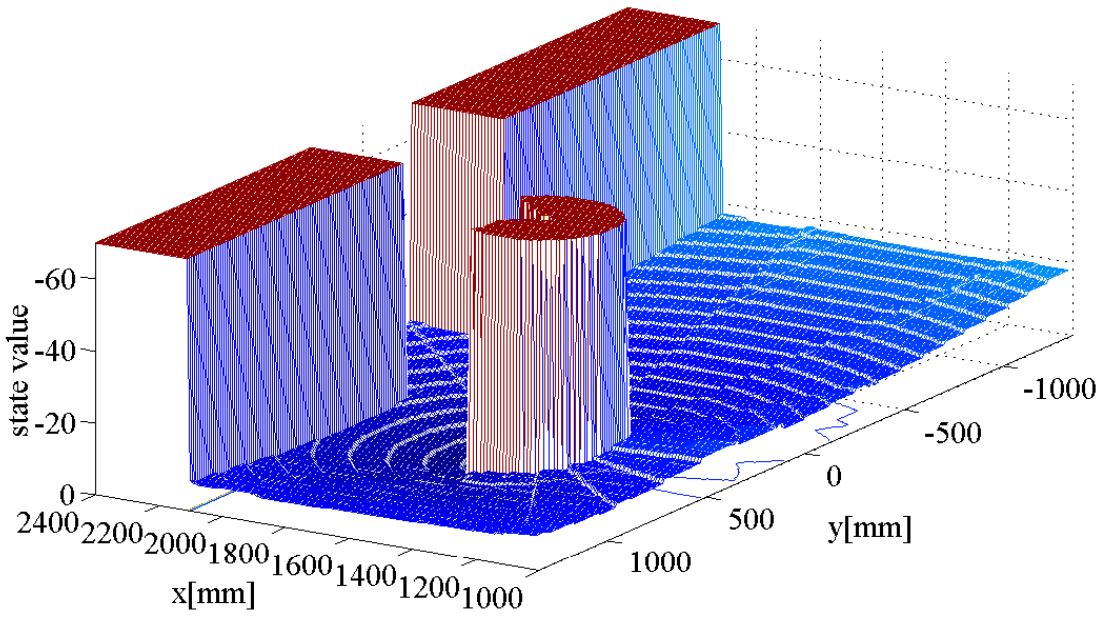

Fig. 4.11 ロボットがボールの方向を向いている場合

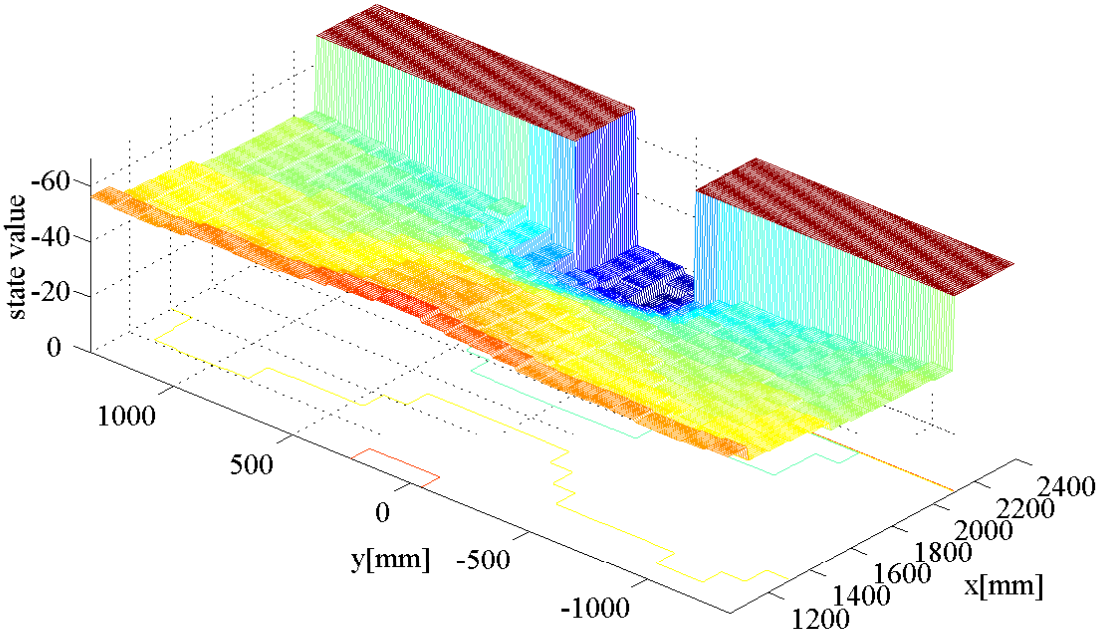

Fig. 4.12 ボール観測なしの場合

ることが分かる．ロボットは，現在位置によって，ゴールに向かうか，ボールに向かうかを選択することとなる．

Fig.4.14 のように，キーパーに最低限のタスクを行わせる方策が得られた．(a) は，ゴールよりもボールまでの歩数が少なくてすむので，ゴールに戻らず，ボールへ向かう経路が選択されている．(b) では，ロボットはボールを見ながら後ろ向きでゴールに戻っている．(c) では，キーパはボールを観測できていないため，自殺点回避のために，ゴールの方向に向きなおしてからゴールに戻っている．(d) では，ボールが近いために，定位置から出てボールに向かっている．このように，動的計画法を用いると，設計者がタスクの性質を指定するだけで，if then else ルールにすると膨大なコードの量になるような行動が計画できる．

定義域外でのキーパーの行動

フォワードと同様，キーパーも Table 4.4 の定義域以外での行動を定義する必要がある．フォーワード同様，ロボットが定義域から外に出たときは，最も近い区間の方策を用いる．φ についても，このルールを適用する．

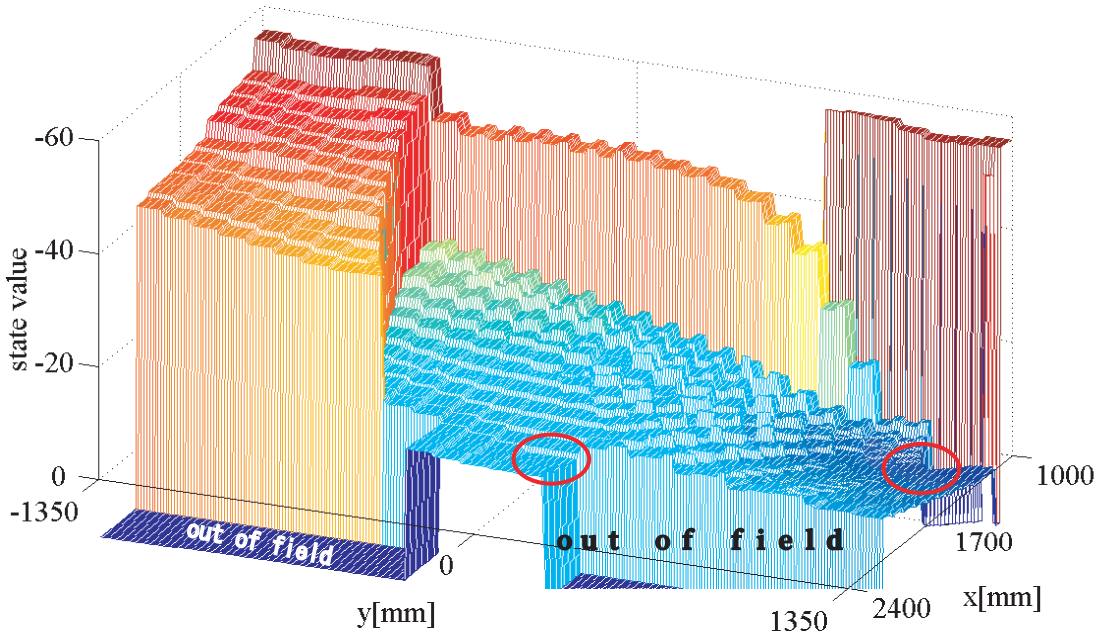

Fig. 4.13　2つの位置に極大値（丸囲みの部分）を持つ場合

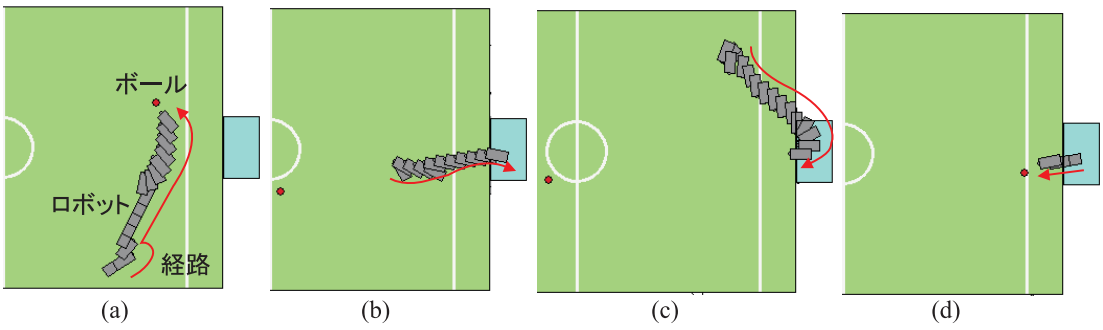

Fig. 4.14　価値反復で得られたロボットの行動

4.4　おわりに

本章では，動的計画法，価値反復の説明を行った．動的計画法は，終端状態が決まっている問題に対して，評価関数（報酬）を最大にする問題（最適化問題）を解く手法の総称である．価値反復は，動的計画法のなかで離散状態空間に対して適用できる手法の一つであり，これを用いて，フォワード，キーパーの行動計画を行った．これらの行動計画は，フォワード（ボールへの接近行動）についてはタスク終了までのロボット歩数を最少にするように報酬が設定された．また，キーパー行動については，より複雑なタスクに対して計画を行った．複数の性質を持つサブタスク（ボールを確保する，ゴールを守備する）を設定し，それらの終端状態に価値の差を与え，また，ボール観測不可能になる行動や，自殺点の恐れのある行動などに負の報酬を与えることで，著者の意図したキーパー行動の方策が得られた．

計画に要した時間は，内部周波数が 1.2GHz の CPU において，

- ボール接近行動：801,900 個の離散状態，40 種類の行動要素に対して 69 分 39 秒
- キーパー行動：2,979,432 個の離散状態，40 種類の行動要素に対して 121 分 12 秒

であった．また，計画では考慮されなかった状態に対して，適当な行動を方策として与え，ロボットがフィールド内で方策を与えられない状況をなくした．

本章で計画したキーパー行動は，次章で提案する行動決定法が適用される．また，フォワード行動については，シミュレーションにおいてキーパーと対戦させるために用いられる．

第5章　曖昧さを考慮した行動決定

5.1　はじめに

本章では，状態推定の出力 $bel(\boldsymbol{x})$ と状態既知を前提とした方策 π から，任意の $bel(\boldsymbol{x})$ に対して行動をひとつ選択する方策 Π を設計する．それをモンテカルロ法の出力と，前章で得られた離散化された状態空間での方策を用いて実装する．

本章は，以下のように構成される．

5.2 節では，状態既知を前提とした方策 π を用いて，状態推定の出力 $bel(\boldsymbol{x})$ から行動を得るための計算方法（方策 Π）を示す．

5.2 推定の曖昧さを考慮した方策の設計

ここでは，推定の曖昧さを考慮した方策を，Uniform MCL と動的計画法で得られた最適状態価値関数 V^* と，最適方策 π^* を用いて設計する．通常（曖昧さを考慮しない場合）は，方策 π^* のみを実機に実装するが，ここで設計する方法は，V^* を主に用い，π^* は補助的に使用される．

現在の状態の確率密度分布 $bel(\boldsymbol{x}) \in \Omega$ が得られているとすると，連続状態空間 $\mathcal{X}$ 中での最適状態価値関数 $V^*(\boldsymbol{x})$ から，現在の状態価値の期待値

$$\bar{V} = \int_{\mathcal{X}} bel(\boldsymbol{x}) V^*(\boldsymbol{x}) d\boldsymbol{x} \tag{5.2.1}$$

が得られる．また，行動 $a \in \mathcal{A}$ を，$\boldsymbol{x}$ の状態を $\boldsymbol{x}' = f_a(\boldsymbol{x})$ に遷移させる関数とみなすと (これは式 (3.3.2) に相当)，現在時刻において，行動 a を行う価値 Q_a の期待値は，行動前後の $\bar{V}$ の差から，

$$\bar{Q_a} = \int_{\mathcal{X}} bel(\boldsymbol{x}) \{V^* \circ f_a(\boldsymbol{x})\} d\boldsymbol{x} - \int_{\mathcal{X}} bel(\boldsymbol{x}) V^*(\boldsymbol{x}) d\boldsymbol{x} \tag{5.2.2}$$

と表される．この値を最大にするように行動決定すると，Ω を状態空間とみなしたときの方策

$$\Pi[bel(\boldsymbol{x})] = \arg\max_a \bar{Q_a} \tag{5.2.3}$$

が得られる．これは，1 ステップ先の状態価値を最大にしようとする方策であるので，グリーディ（貪欲な）方策と言える．また，Π は，1 ステップ先のまでに，観測によって $bel(\boldsymbol{x})$ の変化が起こることを期待しない方策であるとも言える．

グリーディ方策 Π は，1 ステップ先のみを考慮して行動選択するが，これを一般化すると，n ステップ先の状態価値の期待値

$$\bar{V}^{(n)}(a^1, a^2, \ldots, a^n) = \int_{\mathcal{X}} bel(\boldsymbol{x}) \{V^* \circ f_{a^n} \circ \cdots \circ f_{a^2} \circ f_{a^1}(\boldsymbol{x})\} d\boldsymbol{x} \tag{5.2.4}$$

を最大にするように，行動 $a^1, a^2, \ldots, a^n$ を選択するという方法になる．しかし，観測によって $bel(\boldsymbol{x})$ が途中で変化する場合を予測できないため，数ステップ先の状態価値が上式に従うという予測はできない．また，一般的に観測がないと $bel(\ell)$ の分布が次第に広くなるため，最適な行動を選択しても，数ステップ先には，状態価値の期待値が下がっている可能性がある．以上を理由として，本論文では，式 (5.2.3) の方法のみについて扱うこととする．

得られた方策 Π を，Uniform MCL と，離散的な状態価値関数 $V(s)$ を用いて表現する．第 3 章では $\mathcal{X}_{\mathrm{robot}}$ でのみ推定を行っていたが，ここではサンプルが $\mathcal{X}$ 内に分布していると仮定して説明する．ボールの位置が決定論的に求められているため，本節の説明のみでは実装ができないが，これについては実装の段階で対処法を述べる．

式 (5.2.1) は，各サンプルの位置 $\boldsymbol{x}_i$ における状態価値の平均値：

$$\begin{aligned}\hat{V} &= \sum_{\mathcal{S}} Bel(s_i)V(s_i) \\ &= \sum_{\mathcal{S}} \int_{s_i} bel(\boldsymbol{x})d\boldsymbol{x}\ V(s_i) \\ &= \frac{1}{N_{\mathrm{sample}}} \sum_{i=1}^{N_{\mathrm{sample}}} V(s_i)\end{aligned} \tag{5.2.5}$$

で計算できる．次に，式 (5.2.2) については，行動 a をとったと仮定したときの，移動後のサンプル $\xi_{a,j}$ $(j = 1, 2, \ldots, N_{\mathrm{a,sample}})$ を求めて，それらの位置について式 (5.2.5) を計算し，

$$\hat{Q}_a = \frac{1}{N_{\mathrm{a,sample}}} \sum_{j=1}^{N_{\mathrm{a,sample}}} V(s_{a,j}) - \frac{1}{N_{\mathrm{sample}}} \sum_{i=1}^{N_{\mathrm{sample}}} V(s_i) \tag{5.2.6}$$

とできる．式 (5.2.3) は，このまま適用できる．

5.3　キーパー行動への適用と実装

実際に前節の方法を使うためには，実時間性を保証したり，状態空間の境界付近での計算方法やその他の細かい設定を行う必要がある．ここでは，前章で計画したキーパー行動に対して適用し，ロボットに実装を行う上での細かい設定を述べる．

ボールの位置計測結果の利用方法

ボールの位置計測は，決定論的に行っているので，そのまま計測値 (r, φ) を用いる．Uniform MCL のサンプルが存在する空間を，自己位置に関する空間 $\mathcal{X}_{\mathrm{robot}}$ から，状態空間 $\mathcal{X}$ に拡張したときに，サンプル ξ_i $(i = 1, 2, \ldots, N_{\mathrm{sample}})$ の位置ベクトルを $\boldsymbol{x}_i = (x_i, y_i, \theta_i, r_i, \varphi_i)$ と表現すると，全ての i について $r_i = r, \varphi_i = \varphi$ とする．

計算の簡略化

全行動に対して $\hat{Q}_a$ を求めると，計算量が移動に関する Uniform MCL の計算量と全行動数の積になる．この計算を実時間で行おうとすると，サンプル数の上限が小さくなってしまう．そのため，方策 π^* を用いて，計算量を削減することを考える．サンプルの状態ベクトル $\boldsymbol{x}_i$ $(i = 1, 2, \ldots, N_{\mathrm{sample}})$ に対して，方策 $\pi^*(\boldsymbol{x}_i) = \pi^*(\boldsymbol{x}_j)$ $(\forall i, \forall j)$ であれば，そのままその行動を選択する．また，行動が二つ以上選択されたときには，それらの行動に対して，式 (5.2.6) を適用して，$\hat{Q}_a$ が最大の行動を選択する．後者の手続きに関しては，他に $\hat{Q}_a$ がより大きくなる行動 $a' \neq \pi^*(\boldsymbol{x}_i)$ $(\forall i)$ があるかもしれないが，どのサンプル位置 $\boldsymbol{x}_i$ においても方策によって選択されなかった行動 a' が，$\hat{Q}_{a'}$ を最大にする場合はまれであると考えられる．

状態空間の境界付近や価値関数の不連続な部分でのアルゴリズム

ロボットが好ましくない向きでボールに触れる場合の状態価値が $-\infty$ と設定されているので，ロボットの向きが曖昧であると，このような状態になるサンプルが出現して，状態価値の期待値が定義できなくなる．これについては，キーパーはボール確保が最優先ではないため，近づけないように適当に負の大きな状態価値を与える．一方，ボールが観測できる状態から $r < 120[\mathrm{mm}]$ の領域に入ったサンプルについては，ボールを見失った状態での状態価値を与える．

$\max_a \hat{Q}_a < 0$ のときの対処法

最後に，$\hat{Q}_a$ が最大となったとき，それが負の値であるときの行動について実装方法を述べる．このときに関しては，全サンプルが終端状態でなければ，何か行動を起こさなければならないが，何か行動を起こすと状態価値の期待値が下がるというジレンマが起こる．また，いくつかのサンプルが終端状態だった場合には，全サンプルが終端状態となるまで動かなければならないのか，ある割合に達したら静止するかを，決定する必要がある．

キーパーのタスクの性質を考えると，終端状態で静止している必要性は特にない．また，位置が変わらないと，自己位置推定が起こらずに，$\max_a \hat{Q}_a < 0$ の状態から抜け出せなくなるおそれもあるので，ここでは全サンプルが終端状態に達するまでは，常に行動をとりつづけることとする．

以上の設定を反映させたフローチャートを Fig.5.1 に示す．まず，サンプル位置全てについて，終端状態かどうかが調べられる．全サンプルが終端状態にあれば，ロボットは行動しなくて良いので，行動の選択は行われない．そうでなければ，全サンプル位置における方策が調べられる．それが全て同じ行動を選択しておれば，その行動が選択され，そうでなければ，式 (5.2.6)，式 (5.2.3) が適用されて行動決定が行われる．

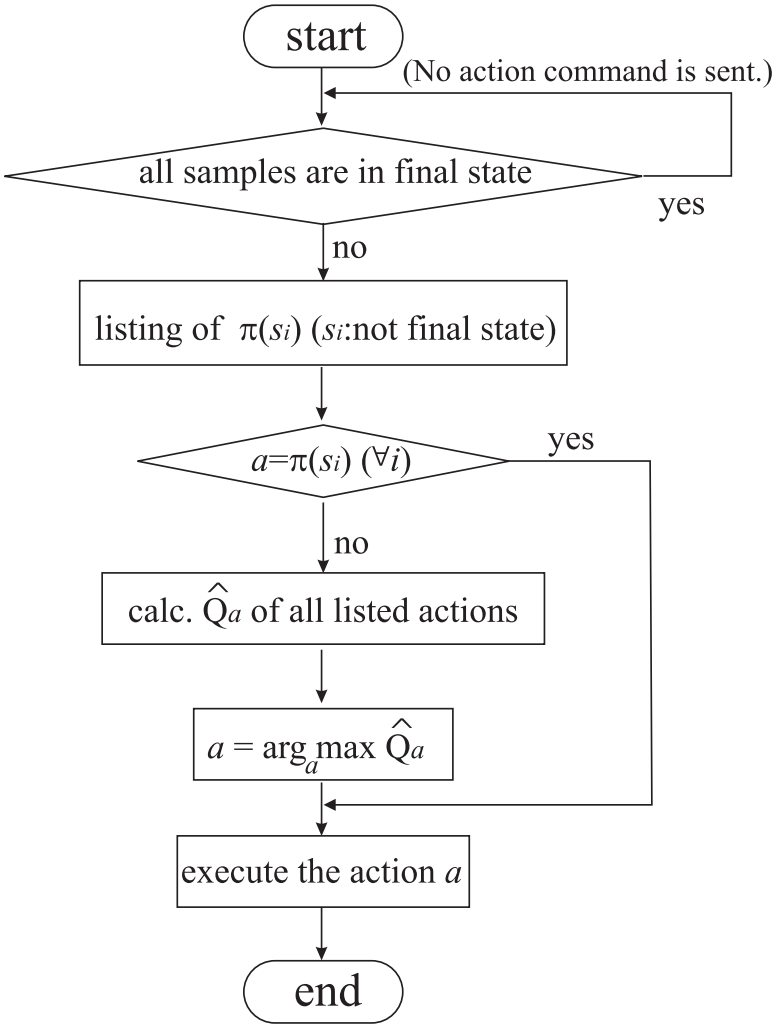
start
(No action command is sent.)
all samples are in final state
yes
no
listing of π(si) (si:not final state)
a=π(si) (∀i)
yes
no
calc. Q̂a of all listed actions
a = arg max Q̂a
a
execute the action a
end

Fig. 5.1　行動決定法

5.4　おわりに

本章では，以下について述べた．

- 曖昧さを考慮した方策 Π の設計
- 方策 Π の Uniform MCL を用いた実装方法の説明
- キーパー行動の行動決定法の設計，実装

ここで設計した行動決定法は，ロボットに最適状態価値関数を実装することで，[横井 01] では不可能な，複数の考えられる行動から，妥当なものを選択することを可能にすることができる．また，[深瀬 02] に比べて，オフライン，オンラインのどちらにおいてもより少ないメモリ使用量で実装ができる．

本章で実装した行動決定法は，次章以降で，シミュレーションと実機実験によって評価が行われる．

第6章　シミュレーション

6.1　はじめに

本章では，浅沼によるロボカップ 4 足ロボットリーグのシミュレータ [浅沼 03] によって，提案した自己位置同定法と，行動決定法の解析，評価を行う．

6.2 節では，シミュレータの説明を行う．

6.3 節では，Uniform MCL のサンプル数と，推定の性能の関係を調べる．

6.4 節では，オンライン時の行動決定方法を，提案手法の他にいくつか用意して，性能の比較を行う．

6.5 節では，本章をまとめる．

6.2　シミュレータ

本章で使用するシミュレータ [浅沼 03] は，筆者の研究グループにおいて，ロボットと同じソースコードのデバッグを PC だけで行って開発の効率化を図ることや，仮想環境でロボットに学習を行わせることなどを目的に開発されている．シミュレータは，クライアント/サーバアプリケーションとなっており，各アプリケーションが TCP/IP によってデータ通信を行う．Fig.6.1 は，ある瞬間のシミュレータの様子であり，図中のウィンドウは，それぞれ以下の役割を持つ．

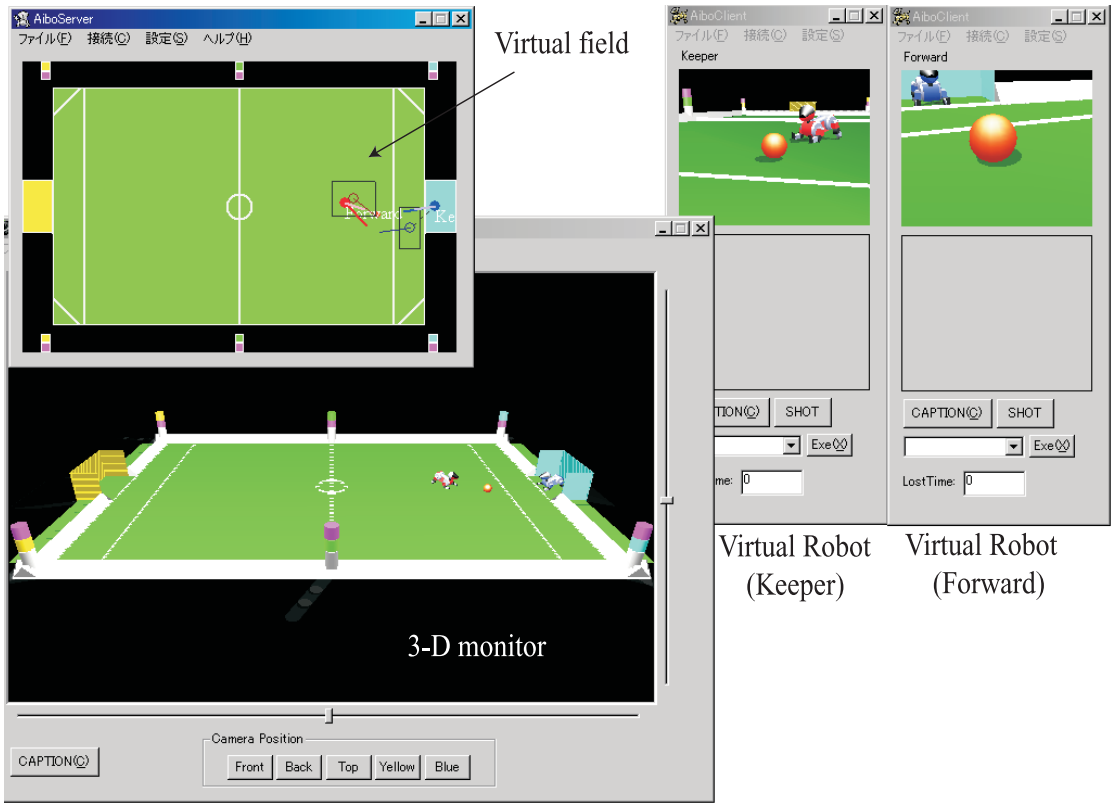

Fig. 6.1　シミュレータ

仮想ロボット（右上）　クライアントアプリケーションで，ひとつのアプリケーションは，サーバと通信する部分と，仮想ロボットの部分から構成されている．仮想ロボットのソースは，ほとんどがロボットに実装されるものと共通になっている．このシミュレータのウィンドウには，仮想ロボットがその位置で得る画像が Open-GL で計算されて表示され，仮想ロボットはこの画像を用いて実機と同様 Fig.6.2 のように，色抽出を行い，画像処理を行う．

仮想フィールド（左上）　各クライアントとデータの受け渡しを行うサーバアプリケーションである．4 足ロボットリーグのフィールド上の物体の形状や位置関係のデータベースを所持しており，このデータベースに基づいて，フィールド上の物体（ボールやロボット）の運動をシミュレートする[*1]．

立体表示部（下）　フィールドの物体の位置を仮想フィールドから受け取り，フィールドの様子を立体的に表示する．

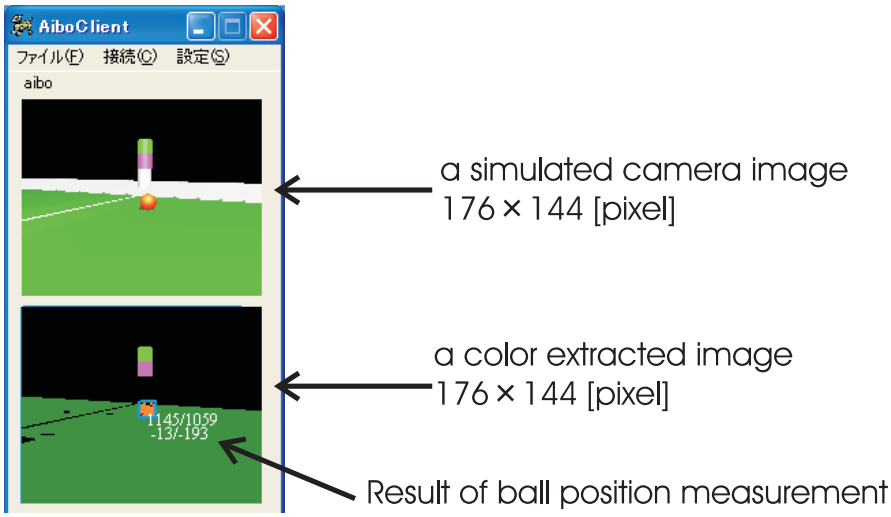

Fig. 6.2　仮想環境での画像処理

[*1] ロボット同士の衝突判定は未実装である．

6.3 Uniform Monte Carlo Localization の評価

6.3.1 推定の性能評価・サンプル数の変化による影響の調査

Uniform MCL の性能と，サンプル数が推定に与える影響を調べる．シミュレータ中で，ロボットに行動させて，歩行後のロボットとサンプルの位置を比較する．ここでは，以下の二つの実験を行う．

- **実験 A: 常にランドマーク観測可能な場合** この実験では，ロボットのカメラは，常にランドマークが観測できるパン角で左右に振られる．また，行動はランダムに選択される．
- **実験 B: タスク中でランドマーク観測に制限がある場合** ロボットは，ボールをゴールに運ぶタスクを行う．ボールが視界に入っている場合は，ロボットは常にボールをトラッキングするため，ランドマークの観測頻度は限られる．

その他の条件を以下に示す．

- 他機は存在しない．
- シミュレーション上で，照明の変化等は起こさない．
- シミュレートされるロボットの歩行には，壁に衝突したとき以外は，雑音をのせない．
- 自己位置が全く分からない状態から開始する．つまり，ロボットは，サンプルは $xy\theta$ 空間全域に拡散した状態から推定を始める．

これらの条件で，ロボットに 300 歩行動をさせる．そして，各歩行後におけるサンプルの分布について，以下のような統計を取って記録する．

- xy 平面での誤差：サンプル位置の x, y それぞれについて平均をとった点と，ロボットの xy 平面での位置との距離
- θ 軸での誤差：サンプル位置の θ の平均値と，ロボットの方向との差
- xy 平面で最も近いサンプルとの距離：xy 平面で，ロボットの位置から最も近いサンプルまでの距離

これらは，前の二つの事項が，ロボットの位置を一点だけを推定する能力の指標であり，三番目の事項が，曖昧さを表現する能力の指標となる．曖昧さが表現できておれば，推定が進んでいない状態でも，サンプルがロボットの付近に存在しているはずである．

また，実験 A のようにロボットに行動させて，シミュレートされた環境内でランドマークの位置計測を行ったときに得られるランドマークの方向の計測値と真値との誤差の度数分布を Fig.6.3 に示す．観測は，800 回行われたが，Fig.6.3 を見ると，シミュレーションではフィールド周囲に試合と関係のない物体がないため，大きな誤差は見られない．また，3.6 節で設定した許容誤差 20[deg] 以内に収まっている．また，ランドマークの幅と，ランドマークとロボットの距離の関係を Fig.6.4 に示す．3.6 節で述べたように，計測される幅とランドマークの距離の関係にはばらつきがあるが，計測される幅と，ランドマークの距離の上限には，一定の規則があることが分かる．ロボットは，このような性質の情報から，自己位置推定を行う．

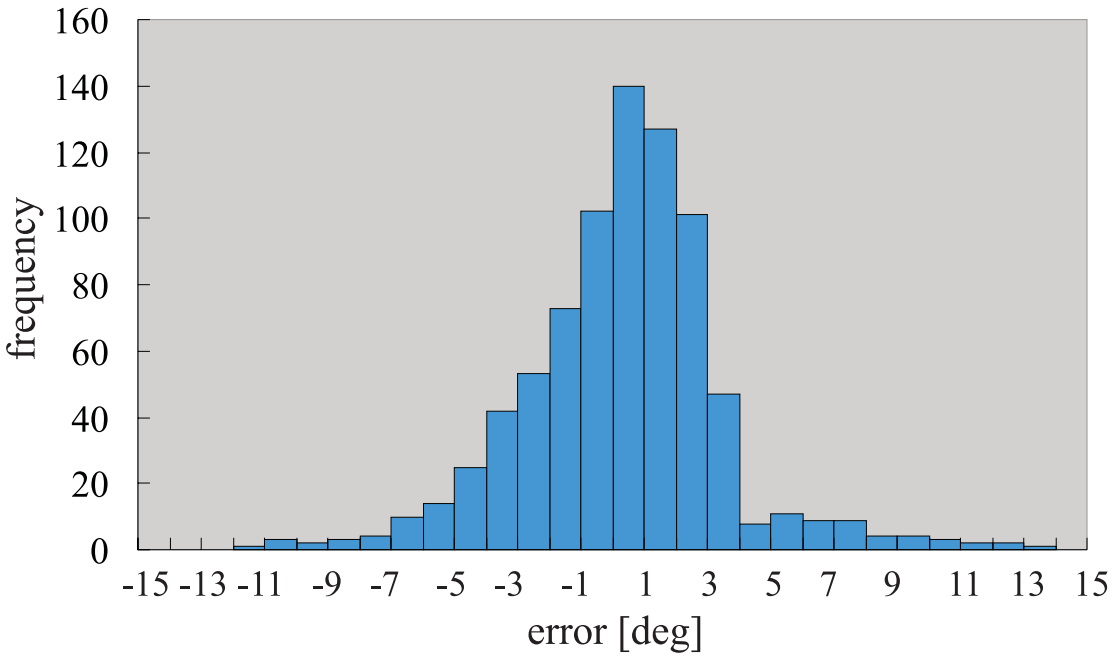

Fig. 6.3　ランドマークの方向計測の誤差

シミュレーション結果を示す．まず，誤差の二乗平均を Fig.6.5,6.6 に示す．Fig.6.5 が xy 平面での誤差で，Fig.6.6 が θ 軸での誤差である．実験 A では xy 平面におい

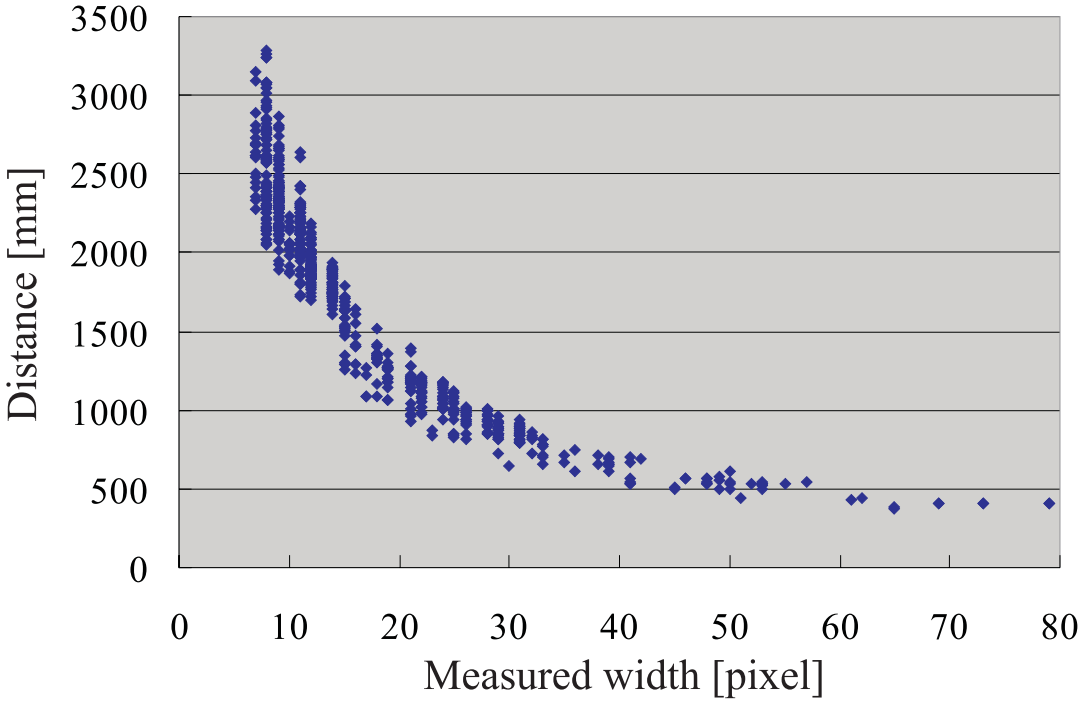

Fig. 6.4 ランドマークの距離と，観測されたランドマークの幅の関係

て，サンプル数が 100 以下のあたりから少しだけ誤差が大きくなっているが，サンプル数が 10 個のときの方が，10000 個のときよりも結果がいいなど，サンプル数が 10 個以上であれば，問題なく推定ができている．したがって，観測が十分可能で，曖昧さの表現が必要なく，正確な自己位置を求めたい場合には，サンプル数は 10 個程度でよいことがわかる．また，実験 B のように得られる情報が少ない場合は，サンプル数が少なくとも 100 以上はないと，安定した推定ができないことが分かる．

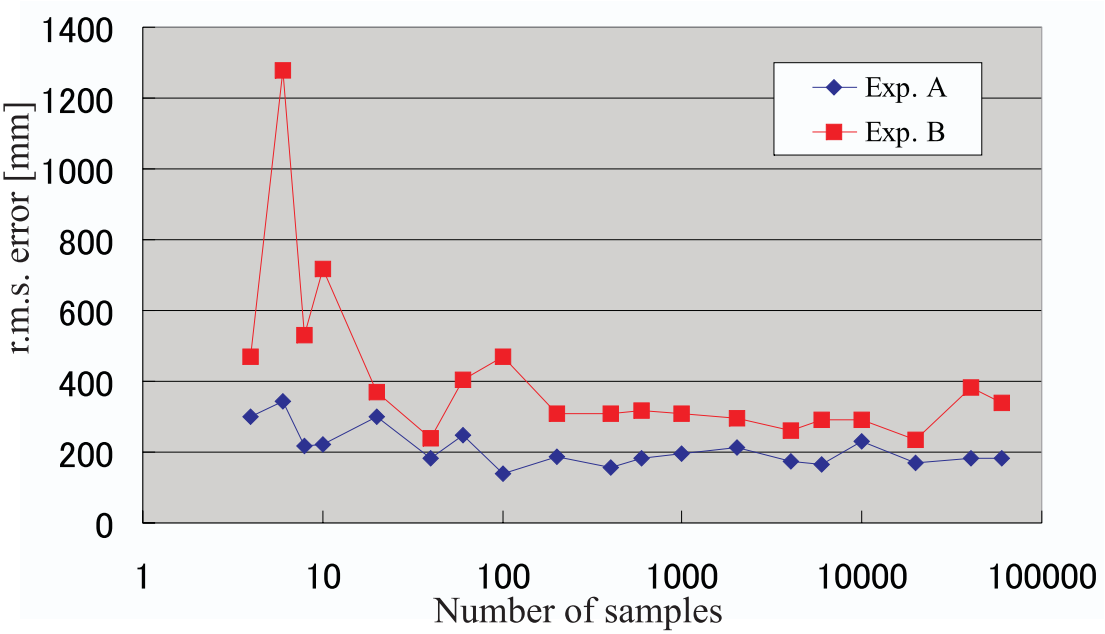

Fig. 6.5　真値と推定値の二乗平均誤差（xy 平面）

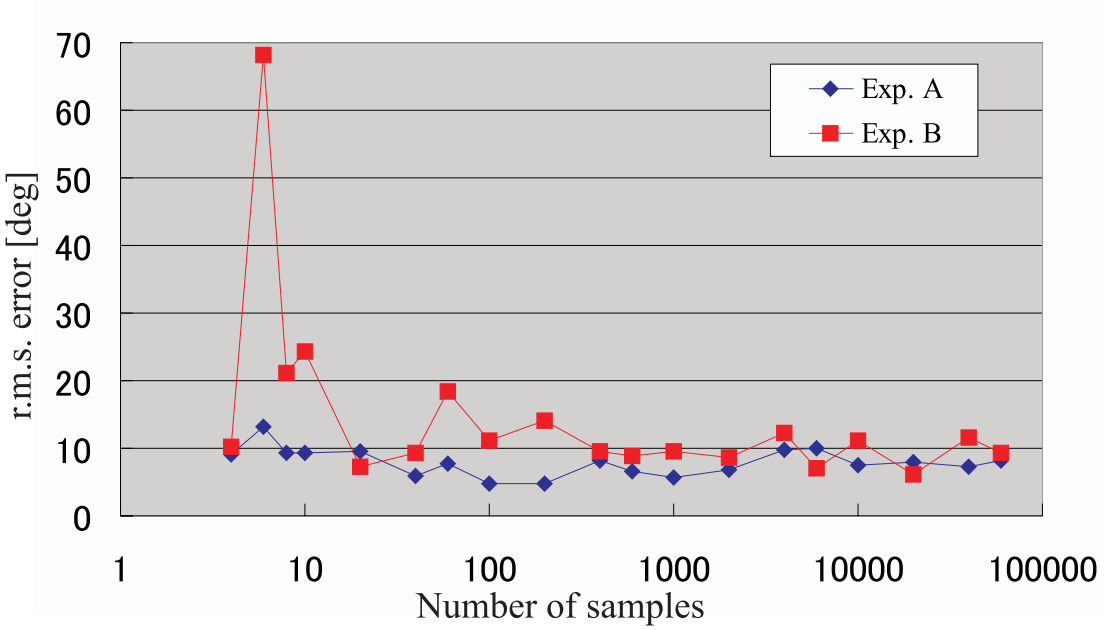

Fig. 6.6　真値と推定値の二乗平均誤差（θ 軸）

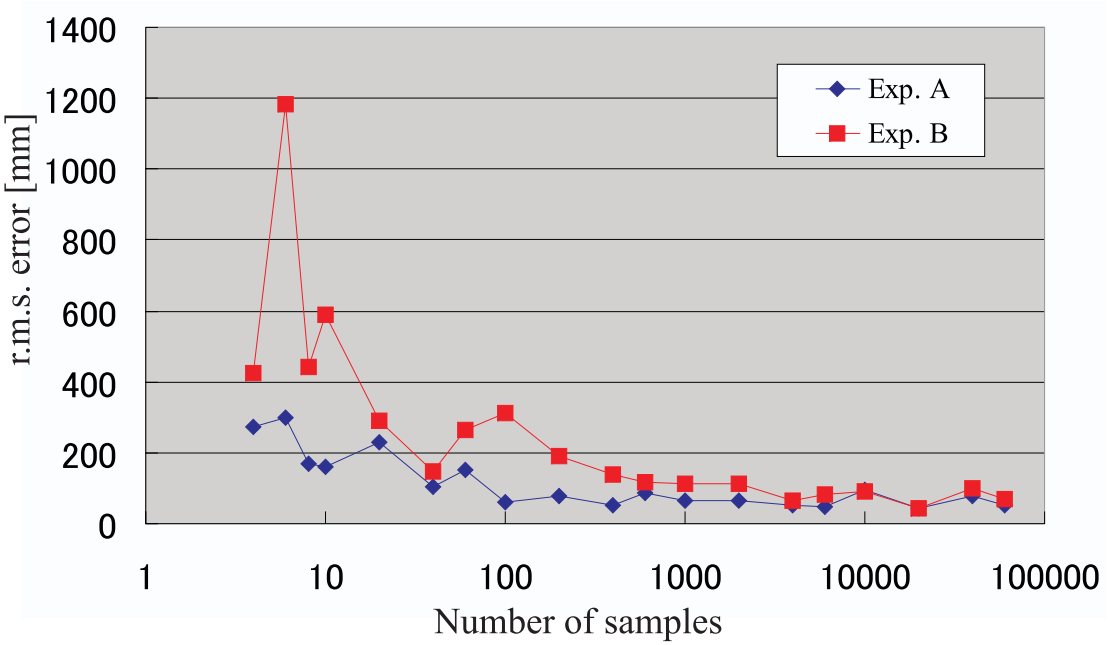

Fig. 6.7 xy 平面で最も近いサンプルとの距離の二乗平均

次に，最も近いサンプルとの距離について，結果を Fig.6.7 に示す．Fig.6.5,6.6 では，実験 A と B で誤差に違いが見られたが，サンプル数が十分（1000 個程度）であれば，Fig.6.7 に見られるように，実験 A と実験 B でのサンプルとの距離に，大きな違いはみられなかった．これは，実験 B のように情報が不足する場合に，サンプルがそれを反映して広く分布するからであると考えられる．

6.4　行動決定法の比較，評価

提案したオンラインでの行動決定手法と，曖昧さを考慮しない行動決定方法でキーパの性能を比較し，提案手法の有効性を検証する．フォワードロボットと，キーパーロボットを一つ実行し，フォワードにキーパーの守るゴールを攻撃させる．評価は，シミュレーション中のキーパーロボットの状態価値の平均値で行う．つまり，ここでは，状態価値関数 V^* に対して，方策 Π がどれだけ忠実に従うのかを調べる．平均値は，動的計画法で考慮しなかった状態になった場合を除いて計算する．

曖昧さを考慮しない行動決定方法では，Uniform MCL のサンプルの位置の平均値をロボットの推定位置として，4 章で計画した方策から行動を決定する．つまり，推定で得られる自己位置の曖昧さは用いない．

シミュレーションの方法を説明する．最初，キーパーとフォワードをフィールド中央に置き，ボールをフォワードが攻める方向の反対側の，フォワードの傍に置く．その後，ゴールが決まるごとにボールのみフィールドの中央に移動させる．つまり，フォワードはゴールの決まったときの位置からボールを取りに行き，再びゴールを攻撃する．キーパーは，その間，方策にしたがってゴールに待機する．フォワードは，曖昧さ未考慮の方法で動作させ，ボールを見失ったときはその場で回転してボールを再発見させる．評価は，真の位置に対して与えられる状態価値の平均値で行う．つまり，方策がどれだけ状態価値関数に対して忠実に行動を選択しているかを評価する．また，ボールがロボット付近で激しく動くと，瞬間的に状態価値が $-\infty$ となってしまうが，このときは，ボールを観測していないときの状態価値を記録することとした．

Table 6.1 に，フォワードが 1000 歩行動する間の，キーパーの状態価値の平均値を示す．提案手法の場合は，曖昧さ未考慮の場合に比べて，状態価値が歩数換算で 1.6 以上良かった．

次に，同様のシミューレーションで，キーパーロボット全体の性能を評価する．今度は，フォワードを 2 台にして，より厳しい状況で守備を行わせる．評価は，ゴールを決めるまでに 2 台のフォワードが費やした歩数[*2]で行う．フォワードは静止しないので，この歩数はキーパーが得点を許すまでの平均時間に比例し，値が大き

[*2] フォワードは静止行動をとらないので，2 台のフォワードの歩数に差は無いはずであるが，サーバとの接続のタイミングで歩数に差が生じる．

Table 6.1 状態価値の平均値

行動決定法	平均値
曖昧さ未考慮	-16.9
提案手法	-15.3
自己位置が既知	-14.5

いほうが，キーパーの性能が高いといえる．また，参考のため，キーパーにサーバから実際の位置を与えるときと，キーパーを置かずにフォワードにゴールを攻めさせたときフォワードの歩数も調べる．

Table 6.2 に，シミュレーションの結果を示す．曖昧さ未考慮の方法が，キーパーなしのときに比べて，1.5 倍程度の時間しか耐えることができなかったのに対し，提案手法では 2.5 倍程度の時間耐えることができた．このように，キーパーとしての性能も，提案手法は，曖昧さを考慮しない方法に比べて有効であることが示せた．

Table 6.2 フォワードの歩数（10 ゴール試行）

行動決定法	歩数（1 ゴール，1 台あたり）
キーパーなし	180
曖昧さ未考慮	269
提案手法	455
自己位置が既知	659

.

6.5　おわりに

本章では，シミュレータによって，提案した自己位置同定法と，行動決定法の解析，評価を行った．主要な結果をまとめると以下のようになる．

- 自己位置推定について
  - ランドマークの方向計測の最大誤差が 15[deg] で，観測が十分なときの自己位置の精度は，精度 200[mm]，8[deg] 程度．
  - 精度 200[mm]，8[deg] 程度の推定で，曖昧さの表現に必要なサンプル数は，1000 個程度．
- 行動決定法について
  - 曖昧さを考慮した方法では，自己位置推定の平均値を用いる方法に比べて，平均すると状態価値が高い状態にロボットが存在する．
  - キーパーの行動計画結果について，提案手法を適用したほうが，自己位置推定の平均値を用いる方法に比べてキーパーとしての性能が高くなる．

第7章　実機実験

7.1　はじめに

本章では，設計，実装した自己位置推定法と，行動決定法を実機により評価する．

7.2 節では，Uniform MCL の評価を行う．まず，理想的な状況や，環境の変化した状況下での推定の性能を調べる．

7.3 節では，提案した行動決定法の評価を行う．この行動決定法から得られる，特徴的なロボットの行動の例を挙げる．また，他の類似する行動決定法を実装し，提案手法との比較を行う．

7.4 節では，本章をまとめる．

7.2 Uniform Monte Carlo Localization の評価

ここでは，Uniform MCL の様々な性質を，実機によって評価する．4 足ロボットリーグのフィールドにおいて自己位置推定アルゴリズムの比較を行った文献 [Gutmann 02] には，このフィールドで自己位置推定を行うときの精度（誤差の絶対値平均）が，100[mm] 前後になることが読み取れる．ただし，ここでは 4.2×2.7[m] のフィールドのランドマーク配置ではなく，3×2[m] の長方形の 4 隅と，長辺の中間にランドマークを配置して実験が行われているので，実際には，よい照明条件のもとで，優れた自己位置推定アルゴリズムの精度が 100[mm] から 200[mm] 程度になることがおおまかに予想される．

7.2.1 定点観測

まず，ロボットが静止した状態で，周囲をカメラで十分観測した後の推定位置精度を調べる．これにより，自己位置が確実に，かつ正確に求められるかどうかを調べる．

実験は，サンプル数の上限を 1000 点とし，次の手順で行う．

(1) 自己位置が全く分からない状態のロボットをある位置，方向に置く．

(2) ロボットは，30 秒間カメラを振り続ける．カメラを，Fig.2.5 の座標で，$(tilt, pan, roll) = (25, 89, 0), (0, 0, 0), (25, -89, 0), (25, 89, 0)$[deg] の順に動かす．動かす周期は，3 秒とする．

(3) ちょうど 30 秒後の推定結果を記録する．

Fig.7.1 に見られるように，位置（直線 $x = 0, 1000, 1750$[mm] と $y = 0, 500, 1000$[mm] の交点の 9 種類），向き（$\theta = \pm 45$[deg] と $\theta = \pm 135$[deg] の 4 種類）を組み合わせた 36 種類の位置，方向にロボットを置いて，それぞれについて一回ずつ上記の実験を行った．これらのロボットを置く位置や方向等は，著者が事前に巻尺，分度器等で慎重に計測したが，2,3[mm]，1,2[deg] 程度の測定誤差がある．また，実験用フィールドにおいては，ランドマークやゴールの位置が固定できないため，これらの位置には 2,3[mm] 程度の位置のずれがあることを記しておく．

評価には，以下の項目を使用する．

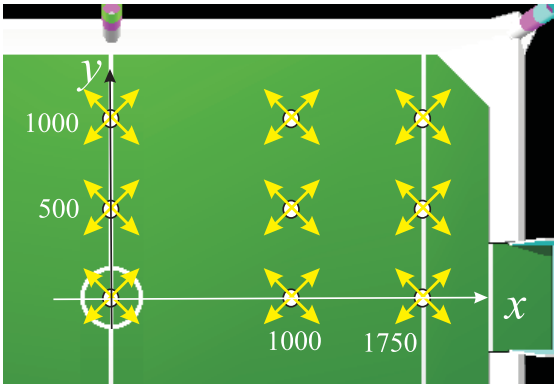

Fig. 7.1 ロボットを置く位置と方向

- r.m.s. (route mean square)：全サンプルの位置，方向の平均値 $(\bar{x}, \bar{y}, \bar{\theta})$ と，ロボットの位置 $(x_{\mathrm{robot}}, y_{\mathrm{robot}}, \theta_{\mathrm{robot}})$ との差に対して二乗平均をとったもの．これは，Uniform MCL が，ロボットの位置一点を推定する能力を示す．
- 最大誤差：全 36 回の試行のうち，$(\bar{x}, \bar{y}, \bar{\theta})$ と $(x_{\mathrm{robot}}, y_{\mathrm{robot}}, \theta_{\mathrm{robot}})$ との差が最も大きかったもの．この値が小さいと，安定した推定ができていることになる．
- 区間の幅の平均：サンプルが存在する領域を x, y, θ の各軸に投影した区間 $[x_{\min}, x_{\max}], [y_{\min}, y_{\max}], [\theta_{\min}, \theta_{\max}]$ を考えたときの，この区間の長さの平均値．この値が小さいと，Uniform MCL が，より断定的に自己位置を推定していると言える．
- 真値が区間内にある確率：全 36 回の試行で，上記の区間内に，ロボットの位置が存在している割合．この値が大きいほど，Uniform MCL は曖昧さを考慮して推定を行っていると言える．
- 区間からの平均距離：全 36 回の試行に対して，ロボットの位置が区間内に存在しているときは 0，それ以外のときは区間の端からロボットの位置までの長さについて，平均をとった値．この値も，曖昧さの考慮の指標となる．

これらは，各軸，xy 平面，$xy\theta$ 空間に対して，(定義できる場合には) 評価される．

実験結果を Table 7.1 に示す．また，推定の性能を直観的に示すため，$\theta = 45$[deg] の場合の，x, y についての推定結果を Fig.7.2 に示す．区間の幅が示すように，自己位置を推定するのに十分な情報が得られなかった場合が多かったにもかかわらず，結果として得られた誤差の二乗平均は，xy 平面上で 233[mm]，θ 平面上で 6.5[deg] と（協調などの複雑な動作を考えなければ）サッカーを行うには十分な精度が得ら

れている．また，36 回の試行中，誤差が最も大きかったものでも 393[mm]，16[deg] と，安定して推定ができている．また，区間中にロボットが存在する確率が，47% 程度と低くなっているが，区間からの平均距離を見ると，真値が区間から出ている場合でも，区間のそばに真値が存在していることが分かる．区間からロボットの位置が外れてしまう理由は，Fig.7.3 のように説明できる．これは，(a) のようにサンプルが分布しているとき，(b) で情報 η が入力され，サンプルを消去しない領域 $L(\eta)$ の境界からロボットの位置が近くなった場合，(c) のように，ロボットの片側のサンプルが全て消去されてしまうということを表した図である．サンプル数が 1000 個程度であると，このような現象がよく起こると考えられる．

Table 7.1 定点観測結果（サンプル数 1000 個）

	r.m.s.	最大誤差	区間の幅の平均	真値が区間内にある確率	区間からの平均距離
x 軸	186[mm]	393[mm]	580[mm]	61[%]	62[mm]
y 軸	141[mm]	375[mm]	395[mm]	58[%]	50[mm]
θ 軸	6.5[deg]	16[deg]	19[deg]	67[%]	2.2[deg]
xy 平面	233[mm]	393[mm]	/	47[%]	/
$xy\theta$ 空間	/	/	/	47[%]	/

7.2.2 定点観測による照明条件の変化による影響の評価

フィールド上の照明の数を変えて，定点観測を行う．フィールドには，4 箇所に 2 本ずつ照明（蛍光灯）が設置されている．これを 1 箇所ずつ消して，7.2.1 節と同じ実験を行う．色抽出のキャリブレーションは，照明数を変えても一切行わない．ロボカップでは，同じ照明装置に対して試合前ごとにキャリブレーション作業が行われるほどであり，これは非常に厳しい条件のもとでの自己位置推定となる．

照明数 3 の時は，実験でロボットを置く場所の上（Fig.7.4 の右上）の蛍光灯 2 本を消灯する．照明数 2 のときは，それに加えて，Fig.7.4 の左下の蛍光灯 2 本を消灯する．照明数 1 のときは，Fig.7.4 の右下の蛍光灯のみを点灯する．

結果を Table 7.2,7.3,7.4 に示す．サンプルの平均位置と真値の誤差は，照明条件が悪くなるにつれて大きくなっている．しかし，区間内に真値がある確率は，照

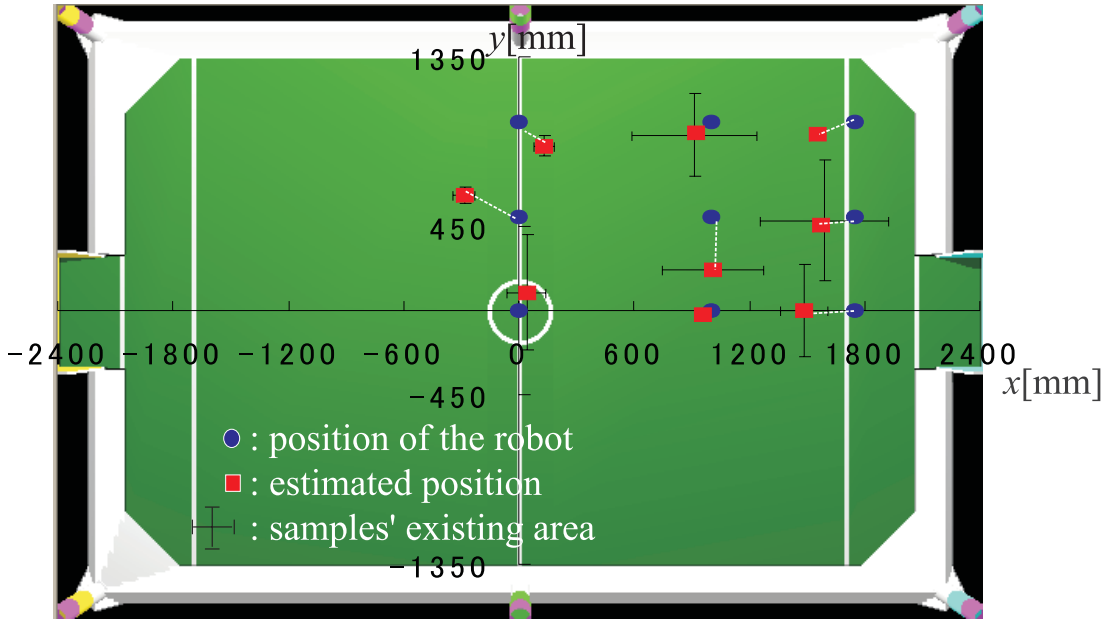

Fig. 7.2 $\theta = 45$[deg] での各位置での推定結果

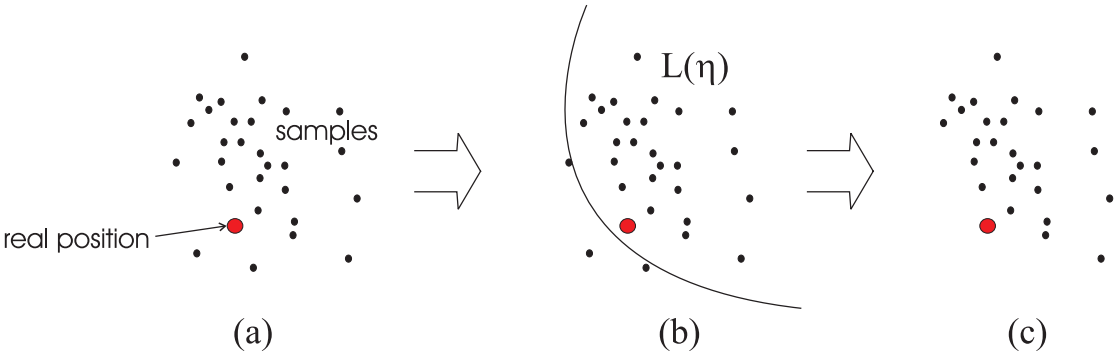

Fig. 7.3 真の位置がサンプル分布から外れる仕組み

明の数に影響を受けていない.

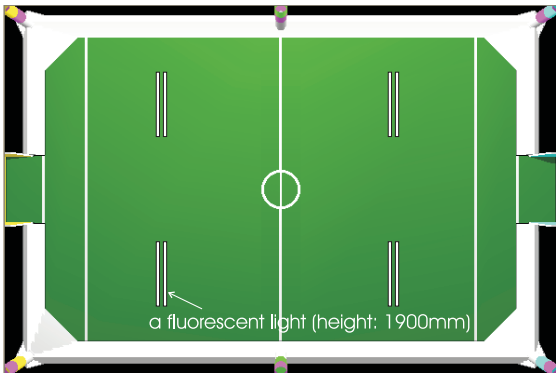

Fig. 7.4 蛍光灯の位置

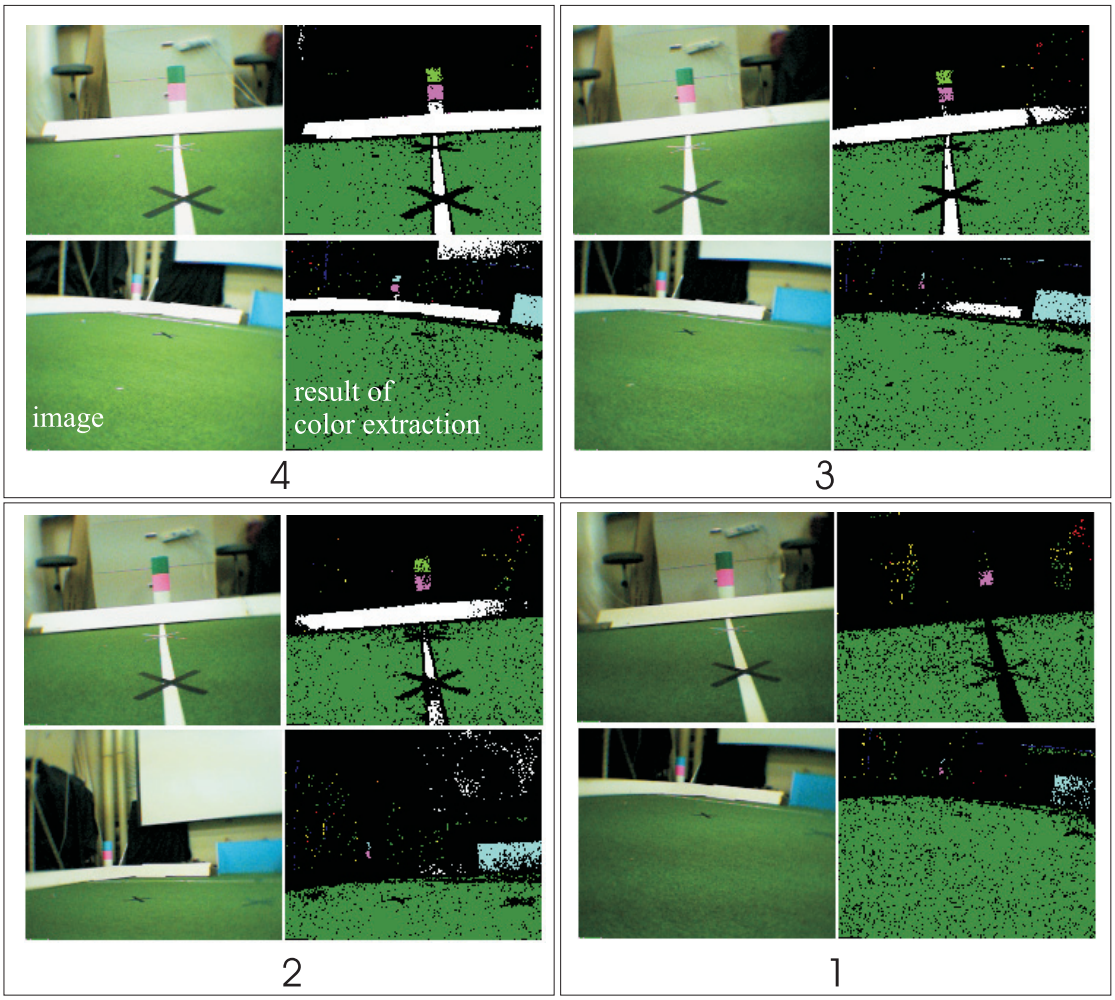

Fig. 7.5 照明を変えて撮影した画像と色抽出結果．数字は照明の数．

Table 7.2　定点観測結果（照明 3 箇所，サンプル数 1000 個）

	r.m.s.	最大誤差	区間の幅の平均	真値が区間内にある確率	区間からの平均距離
x 軸	228[mm]	589[mm]	638[mm]	56[%]	76[mm]
y 軸	174[mm]	496[mm]	440[mm]	50[%]	62[mm]
θ 軸	8.6[deg]	25[deg]	21.1[deg]	58[%]	3.2[deg]
xy 平面	287[mm]	623[mm]	/	47[%]	/
$xy\theta$ 空間	/	/	/	47[%]	/

Table 7.3　定点観測結果（照明 2 箇所，サンプル数 1000 個）

	r.m.s.	最大誤差	区間の幅の平均	真値が区間内にある確率	区間からの平均距離
x 軸	314[mm]	891[mm]	784[mm]	69[%]	85[mm]
y 軸	279[mm]	1127[mm]	656[mm]	78[%]	81[mm]
θ 軸	10.8[deg]	36[deg]	28.9[deg]	78[%]	2.5[deg]
xy 平面	420[mm]	1169[mm]	/	61[%]	/
$xy\theta$ 空間	/	/	/	61[%]	/

Table 7.4　定点観測結果（照明 1 箇所，サンプル数 1000 個）

	r.m.s.	最大誤差	区間の幅の平均	真値が区間内にある確率	区間からの平均距離
x 軸	319[mm]	824[mm]	445[mm]	53[%]	71[mm]
y 軸	335[mm]	1004[mm]	326[mm]	58[%]	88[mm]
θ 軸	12.9[deg]	33[deg]	16.1[deg]	53[%]	3.6[deg]
xy 平面	462[mm]	1299[mm]	/	42[%]	/
$xy\theta$ 空間	/	/	/	39[%]	/

7.2.3　歩行しながらの推定性能

ロボットを歩行させながら自己位置推定を行う．ロボットに 20 回，ランダムに選択された位置，向きから前進歩行（DribFF）をさせて，最後の歩行が終わった時点での推定結果を調べて，真値との比較を行った．カメラの動かし方は，定点

観測のときと同様にする．歩行している時間は，約 14 秒であるので，ロボットは 4 周期半，カメラを振ることになる．

20 回試行した結果を Table 7.5 に示す．Table 7.1 と比較すると，二乗誤差平均は，定点観測より 100[mm]，7.5[deg] 程度悪くなっているが，最大誤差が示すように，どの試行においても安定して推定が行われていることが分かる．

サンプルの分布する区間が x, y, θ ともに狭くなっており，$xy\theta$ 軸全てにおいて区間内に存在する確率は，10%となった．これは，移動によってロボットとランドマークの位置関係が変化し，定点観測のときよりも入力される情報が多様になり，Fig.7.3 で説明した現象が多く起こるからであると考えられる．このことから，サンプル数 1000 個では，情報が十分与えられた場合に関しては，曖昧さの考慮という点において困難が生じると言える．

Table 7.5 歩行しながらの推定結果（サンプル数 1000 個，20 回試行）

	r.m.s.	最大誤差	区間の幅の平均	真値が区間内にある確率	区間からの平均距離
x 軸	207[mm]	500[mm]	215[mm]	20[%]	100[mm]
y 軸	260[mm]	449[mm]	239[mm]	45[%]	130[mm]
θ 軸	14.0[deg]	27[deg]	14.8[deg]	40[%]	5.6[deg]
xy 平面	333[mm]	510[mm]	/	15[%]	/
$xy\theta$ 空間	/	/	/	10[%]	/

7.2.4 推定方向のずれの修正

ここでは，ロボット同士が衝突して，ロボットの方向がずれてしまった状況から，どのように推定位置が更新されるかを調べる．この性能は，式 (3.4.4) の更新則の役割が，重要となる．この手続きが機能すれば，観測によってサンプルの分布が広くなったり，狭くなったりして，次第にずれた後のロボットの位置，方向に分布が収束していくはずである．つまり，途中で必要以上に分布を広くせずに，分布を修正できることが期待できる．

方法は，$x = 1000$[mm]，$y = 1000$[mm] の地点に，ロボットを最初 $\theta = -90$[deg] に置き，2 分程度，十分に自己位置を推定させる．そして，ロボットのカメラを

覆って角度を $\theta = -135$[deg] に置き直して，その後 30 秒間，定点観測を行う．この観測の条件は，7.2.1 節と同じにする．

5 回試行を行った結果を示す．まず，サンプルの向き θ_i $(i = 1, 2, \ldots, N_{\mathrm{sample}})$ の平均値 $\bar{\theta}$ の推移を示す．グラフ上の点は，観測によってサンプルの分布が変化した時のものである．このグラフでは，ロボットが向きを変えられる前の $\bar{\theta}$ の値は，-1[s] にプロットしてある．これを見ると，どの試行でも，観測を開始した直後に 10[deg] 程度の向きの修正が起こり，その後，しばらくした後に，再び 10[deg] 程度の修正が起こっている．例えば，trial 3 では，観測開始からすぐに $\theta = -110$[deg] となり，15[s] 後に突然 $\theta = -120$[deg] となっている．これは，ロボットの置かれた場所から，自己位置推定に使用できるランドマーク，あるいはゴールが少なくとも二つ観測でき，そのうちの一つは，観測しにくく，計測可能な画像が（偶然的に）得られるまでに，時間がかかったからであると考えられる．

また，どの試行でも，30[s] 後も 10-15[deg] 程度の誤差が残った．これは，ランドマークやゴールの位置計測での誤差の許容範囲が片側で 20[deg] と広いため，ロボットと少しだけ θ の異なるサンプルが消去されずに残った結果であると考えられる．

さらに，θ が修正されている間の，xy 平面におけるサンプルの平均位置と，実際のロボットの位置との誤差の推移を Fig.7.7 に示す．試行によっては，平均位置が途中で大きくずれてしまうことがあるが，全体として，ロボットの位置の周りにサンプルの平均位置がとどまっている．また，このときの，x 軸に関するサンプルが存在する区間長の推移を Fig.7.8 に示す．Fig.7.8 からは，Fig.3.9 で説明したような，更新則 Ⅱ の処理が起こったことが読み取れる．例えば，trial 3 では，2 回目の大きな修正の起こった 15[s] 後に，サンプルの分布が広くなって，その直後に急激に狭くなっている．これは，画像処理結果とサンプルの分布の矛盾から，サンプルの分布が Fig.3.9 のように膨張し，次の画像が入力されたときに，その画像処理結果と矛盾するサンプルが消去された結果である．

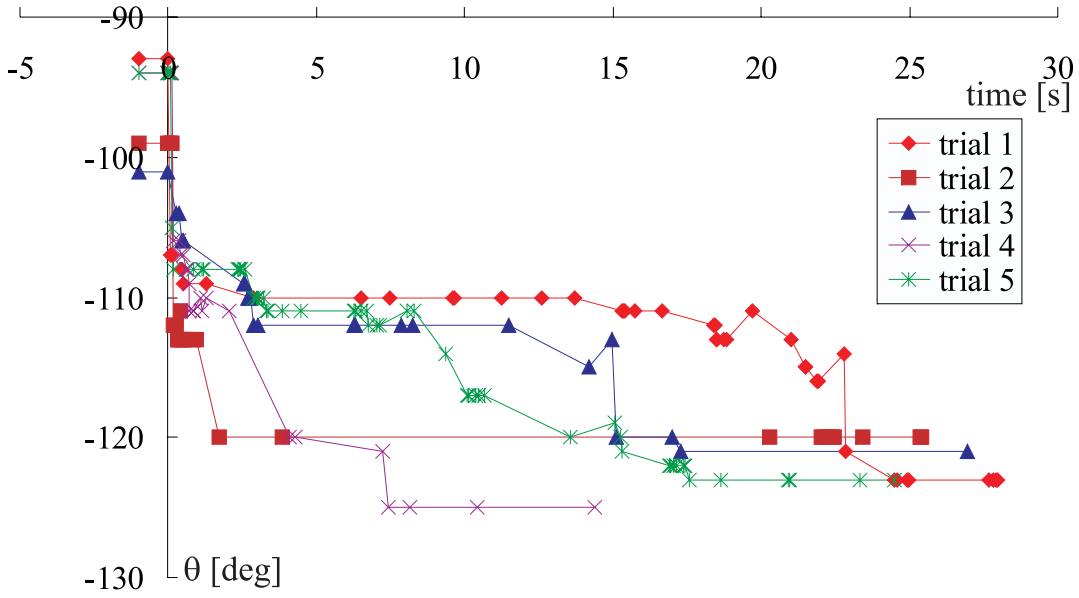

Fig. 7.6 $\bar{\theta}$ の推移

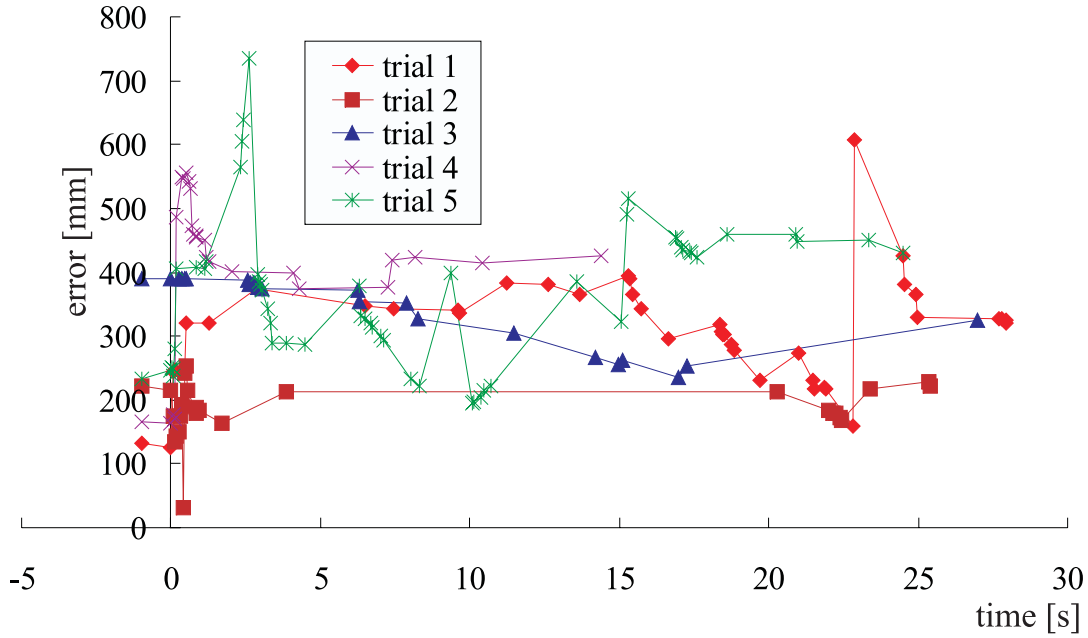

Fig. 7.7 xy 平面での誤差の推移

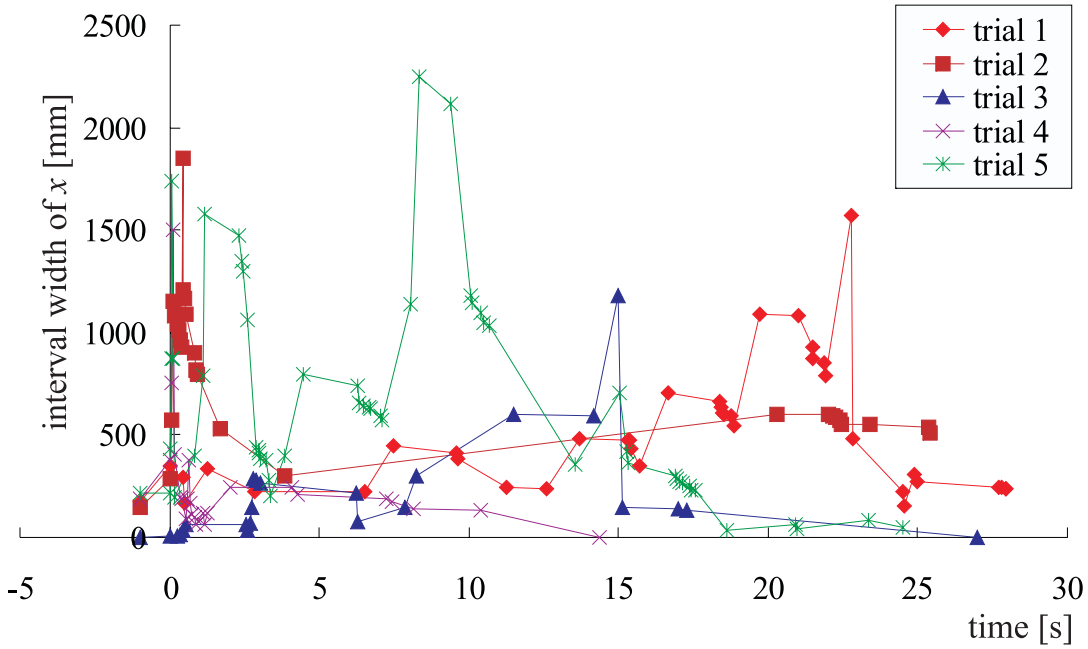

Fig. 7.8 サンプルが存在する区間長の推移（x 軸において）

この結果から，小さな方向のずれに対しては，ずれが生じた直後に観測すれば，位置情報を無駄にせずに修正を行うことができることが分かった．ただし，完全な修正のためには，ロボットがしばらく歩行して，観測する必要があると考えられる．

7.2.5 Kidnapped robot problem

人間がロボットを大きく動かした後，位置が推定されるかを調べる．これにより，Kidnapped robot problem に対する Uniform MCL の性能を明らかにする．

方法は，ロボットを黄色ゴール付近に置いて 1 分程度の時間，位置推定させた後に，ロボットのカメラを覆って $(x, y, \theta) = (1000[\mathrm{mm}], 1000[\mathrm{mm}], -135[\mathrm{deg}])$ に移動し，30 秒間定点観測を行うというものである．

5 回試行を行った結果を示す．まず，xy 平面上で，どのようにサンプルの平均位置が推移したかを Fig.7.9 に示す．グラフ上の点は，観測によってサンプルの分布が変化した時のものである．この図からは，もとの位置から移動後の位置へ，推定位置が観測の度に移動していく様子が見られる．また，Fig.7.10 は，サンプル位

置の x 軸における平均値の推移をグラフにしたものであるが，これを見ると，1[s] から 3.5[s] 程度でロボットの位置 $x = 1000$[mm] の周辺にサンプルが移動していることが分かる．つまり，Fig.7.9 の移動の大部分は，数秒のうちに起こっていることになる．ロボットがカメラを動かす周期が 3[s] なので，サンプルがロボットの位置周辺に移動させるには，一通り周囲を見渡すだけでよいことが分かる．

7.2.4 節の結果も合わせると，ロボカップのフィールドのような長方形の単純な形状の環境では，式 (3.4.4) の手続きが，ロボットが直接知覚できない移動に対して，有効に機能することが示せた．

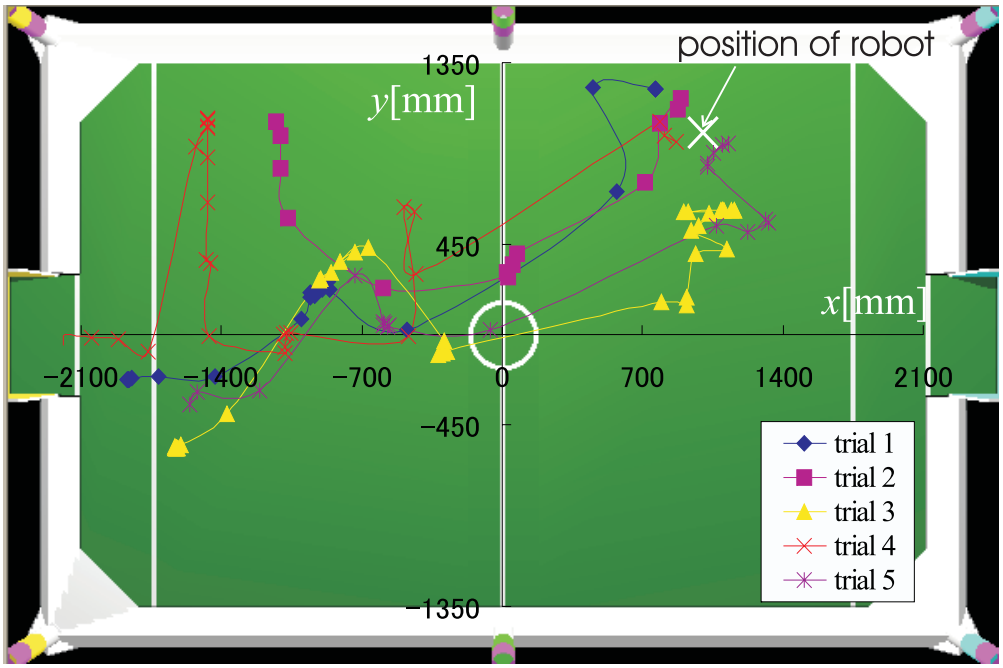

Fig. 7.9 サンプルの平均値の推移

7.2.6 計算量

サンプル数の上限が 1000 個の時について，センサ情報の処理と，移動時のサンプル処理の両者について，処理の平均時間を調べた．センサ処理では，ランドマークを計測する時間も含めて時間を計測したところ，平均 7[ms] であった．また，ロボット移動時にサンプルの位置更新に要した計算時間は，7.5[ms] であった．4 足ロボットリーグでは高速であると言われている SRL（Sensor Resetting Localization）との比較を，Table 7.6 に示す．

この表を見ると，CPU 周波数はことなるものの，その比を大幅に超えて Uniform

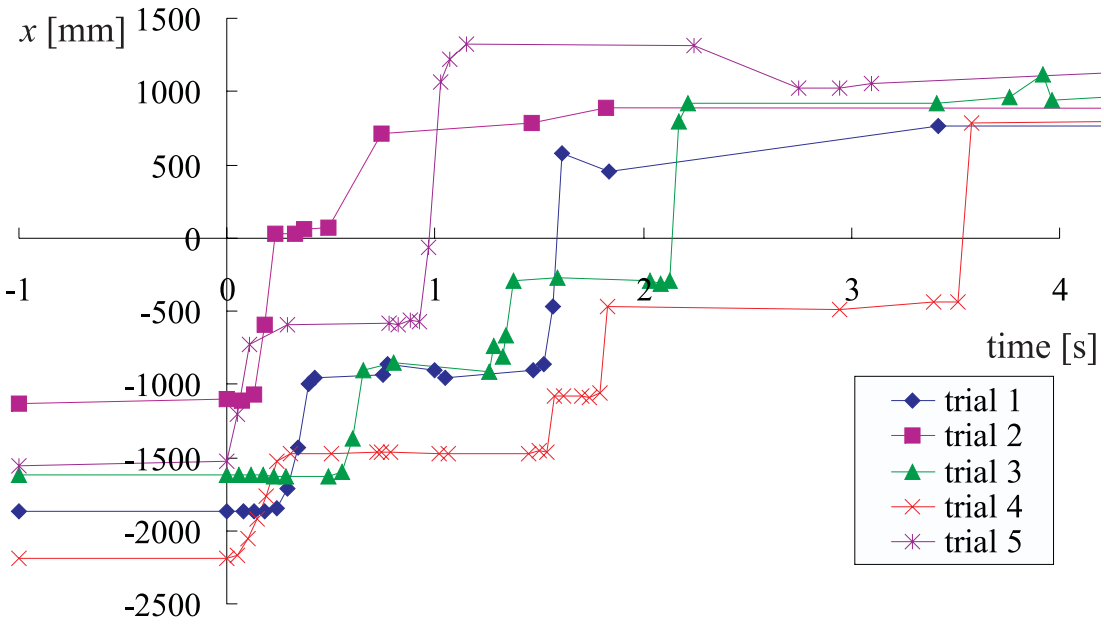

Fig. 7.10 x の平均値の推移

MCL が非常に高速にサンプルを処理できることが分かる．また，画像処理で，複雑なアルゴリズムで高精度に情報を抽出せず，簡単でロバストな情報をすばやく得て，それを重ねあわせて推定を行っていることも，高速化につながっている．画像処理では，全画素に対する演算は，色抽出画像を作成するときの一度だけであり，これに要する計算時間が 1[ms] である．

したがって，Uniform MCL は，低レベルの情報をすばやく重ね合わせて，結果としてよい推定結果を得るという，フィルタとして求められる性能を有していると言える．

Table 7.6 計算量の比較

	CPU	サンプル数	観測時	移動時
SRL [Lenser 00]	MIPS 100MHz	200※	実時間	記述なし
SRL [横井 01]	MIPS 100MHz	200	400[ms]	400[ms]
Uniform MCL	MIPS 192MHz	500-1000	7.0[ms]	7.5[ms]

※: 実時間で計算するため，400 個のサンプルを 200 個ずつ処理している．

7.3　行動決定法の評価

7.3.1　環境変化を反映したサブタスクの選択

シミュレーションにおいては，提案した行動決定手法が有効に機能することを示したが，ここでは実機において，提案した行動決定法が，曖昧さを考慮することによって，環境の変化に柔軟に対処できることを示す．キーパー行動には，複数のサブタスクが一度に計画されているが，提案手法では，ロボットがこのサブタスクを，情報の曖昧さを基準として選択することが期待できる．ここでは，提案手法と，サンプル位置の平均値と方策 π を用いて行動決定する手法を実験で比較することで，サブタスクの選択がどのように行われるかを検討する．

ボールを $x = 1750[\mathrm{mm}]$，$y = 1350[\mathrm{mm}]$ の地点に置き，ロボットを，自己位置を推定していない状態で，$x = 1000[\mathrm{mm}]$，$y = -1350[\mathrm{mm}]$，$\theta = 0[\mathrm{deg}]$ の地点から行動させる．すると，Fig.7.11 のように，ロボットは推定された自己位置の微妙な違いで，ボールを確保したり，ゴールに帰還して待機するようになる．自己位置推定の出力を変質させるために，照明条件を変えたり，ゴール（正確にはゴール板，Fig.2.3 参照）を取り外したりして環境を変える．照明条件を変えれば，フィールド全域において自己位置推定結果が曖昧になり，ゴールを取り外せば，ゴール周辺での自己位置推定結果が曖昧になる．これらに対して，ロボットがどちらのサブタスクを選択するかを集計する．

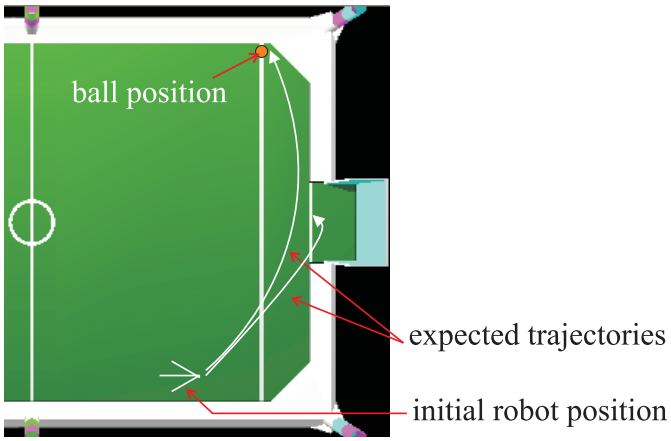

Fig. 7.11　ロボットの初期位置と期待されるロボットの行動

照明条件は，理想的な条件と，照明を一つ消した暗い条件を試す．Uniform MCL

の実験で示したように，照明が暗いと，自己位置の x, y に関する情報の曖昧さが増し，θ については，曖昧さが増すものの，x, y に比べてその増加は大きくない．つまり，ゴールへの帰還は難しくなるが，ボールに正しい位置から近づくことは，それほど難しくはならない．また，ゴール板を外したときも，同様にゴールへの帰還が難しくなると考えられる．

また，タスクの選択は，ボール確保についてはボールに触れた時点，ゴール待機については 5 秒以上ゴール内に脚が 2 本以上入った状態が続いたときに，選択成立とした．また，ロボットがゴール外で 10 秒間，終端状態と間違えて歩行を止めてしまったときは，ゴール待機を選択して，失敗したとみなした．

結果を Table 7.7 に示す．まず，全体的にみると，曖昧さ未考慮の方法が，ボールに向かう頻度が高かった．自己位置の曖昧さ未考慮の方法では，ゴールに帰還しようとしたロボットが，ゴール周辺の壁にぶつかって，なかなかゴールに入れず，あるときボールに近づいて，そのままボールのほうに向かうという現象が多かったからである．また，これと関連して，曖昧さ未考慮の方法は，ゴール板が無いほうがゴール帰還を選択する割合が高かった．これは実験に用いたフィールドが，ゴール板がない状態の方がゴール入り口が 100[mm] 程度広くなるように組み立てられているからである．つまりこれも，曖昧さ未考慮の方法では，壁に衝突する頻度がいかに多かったかを示すものである．一方，提案手法では，あまり壁に衝突しない代わりに，ゴールに何度も出入りする行動が見られた．これは，サンプル全てが終端状態にならないと，ロボットが歩行を止めないからであるが，キーパーの動きとしては問題ではない．

また，提案手法では，照明条件がよい場合に，ゴール板がある場合とない場合で，あまり行動選択に違いが見られなかった．これは，ゴール付近にロボットが達するときに，ゴール帰還に必要な自己位置の推定精度が得られており，しかも壁に衝突して推定に悪影響が出ることが少なく，ゴールの位置計測が必要なかったからであると考えられる．ただし，ゴール板がないとゴール入り口が広くなることが，この結果にも影響しているとも考えられる．一方，照明が暗い状態では，ゴール板の有無で，タスク選択に傾向の違いが見られた．ゴール板があると，照明条件が悪くてもゴール付近では，ゴール板が十分大きな領域で色抽出できるので，ゴールに帰還しやすいが，ゴール板がないと自己位置情報が行動範囲全域で得にくくなるので，ボール確保に向かう頻度が多くなった．また，このときのボール確保のためのロボットの軌道は，他の条件では Fig.7.11 のように，ゴール寄りになったり，一度ゴールに入ってからボールに向かうものが多かったのに対し，こ

の条件ではボールに向かって一直線に向かって，ボール手前で向きを変えるというものが多かった．

Table 7.7 サブタスクの選択結果（単位: %，各 20 回試行）

		提案手法		曖昧さ未考慮		
ゴール板	照明	ゴール待機	ボール確保	ゴール待機 ※	ボール確保	失敗
有	4 個	60	40	35	65	0
無	4 個	55	45	50	50	10
有	3 個	65	35	35	65	10
無	3 個	40	60	60	40	15

※ 失敗を含む

この実験からは，提案手法が，曖昧さを考慮することで，環境の変化に応じてサブタスクの選択傾向を変えることを確認できたと言える．

7.3.2 計算量

ここでは，提案した行動決定法が計算に要する時間を計測する．カメラ画像を含め，様々なセンサ情報の同期をとるには，カメラのリフレッシュレートである 40[ms] 以内で処理する必要がある．ただし，これを満たさなくても，ロボットは動作し，経験的に 100[ms] 以内であれば，ロボットの行動が遅くなるなどの支障は発生しない．そこで，Uniform MCL と，その他の処理に費やす時間を 10[ms] として，好ましい計算時間を 30[ms] 以内，最低条件を 90[ms] として設定する．

計測方法はごく簡単で，キーパーロボットに提案手法で行動させて，ロボットのタイマ関数によって，行動決定処理の呼び出し前後の時間差を [μs] 単位で出力した．また，時間とともに，全サンプルが方策 π^* を用いて選択した行動の種類数をともに出力した．このときに用いた Uniform MCL のサンプル数の上限は，1000 個である．

結果は，592 回の行動決定機会において，計算時間の平均が 19.8[ms]，このとき選択された行動の種類数の平均が 7.08 個であった．平均的には，目標とする 30[ms] 以内で処理が行われている．30[ms] 以内に処理が終わっている場合は，全体の 75[%] であった．

また，行動の種類数と，それらの時の計算時間の平均と，最大値を Fig.7.12 に示す．行動の種類数が 12 個以上の部分で，種類数と時間の関係が右上がりになっていないのは，平均や最大値を安定して得るだけの頻度がなかったからである．目標となる時間 30[ms] 以内で確実に処理ができるのは，サンプルによって選択された行動が 6 種類以内のときであった．また，最低条件の 90[ms] を超えたことは，一度もなかった．

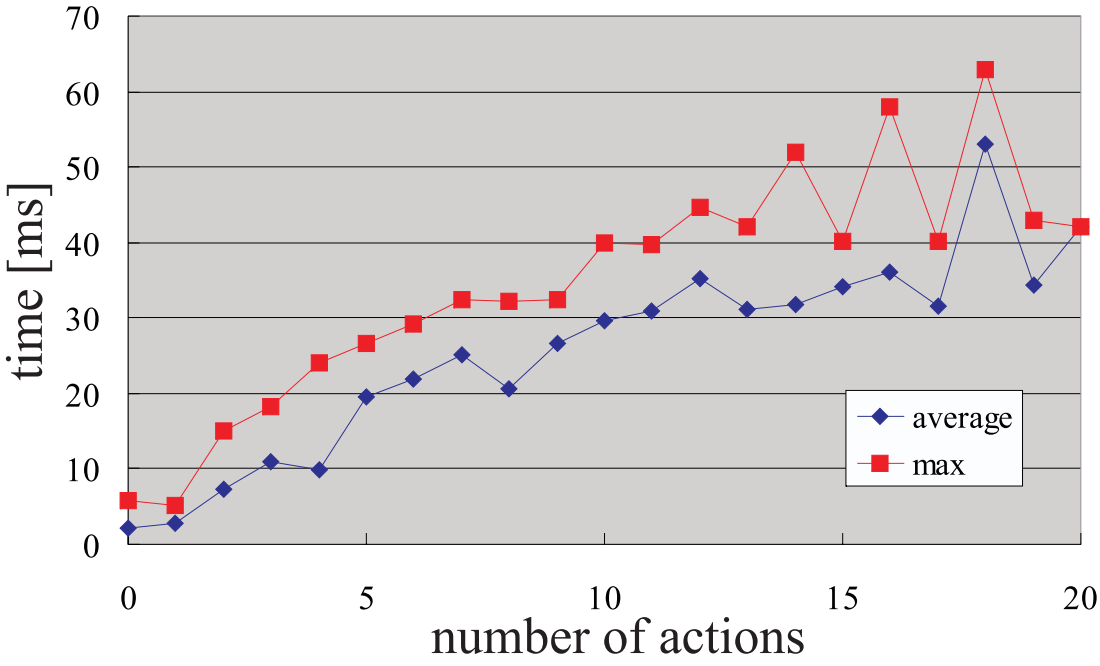

Fig. 7.12 考慮された行動の種類数と計算時間の関係

7.3.3 サッカー行動の例

ここでは，提案した手法を実装したサッカーロボットが実際に動作している様子を示す．まず，Fig.7.13 は，ボールアプローチ行動の方策に基づいて，ロボットが行動した例である．この例では，ロボットは Uniform MCL のサンプルの平均位置を用いて行動決定している．

Fig.7.14 は，キーパーがゴールへ帰還する様子である．この例は，提案した行動決定法でロボットが動作している．曖昧さを考慮していても，このように無駄のない軌道でロボットが移動することが可能である．

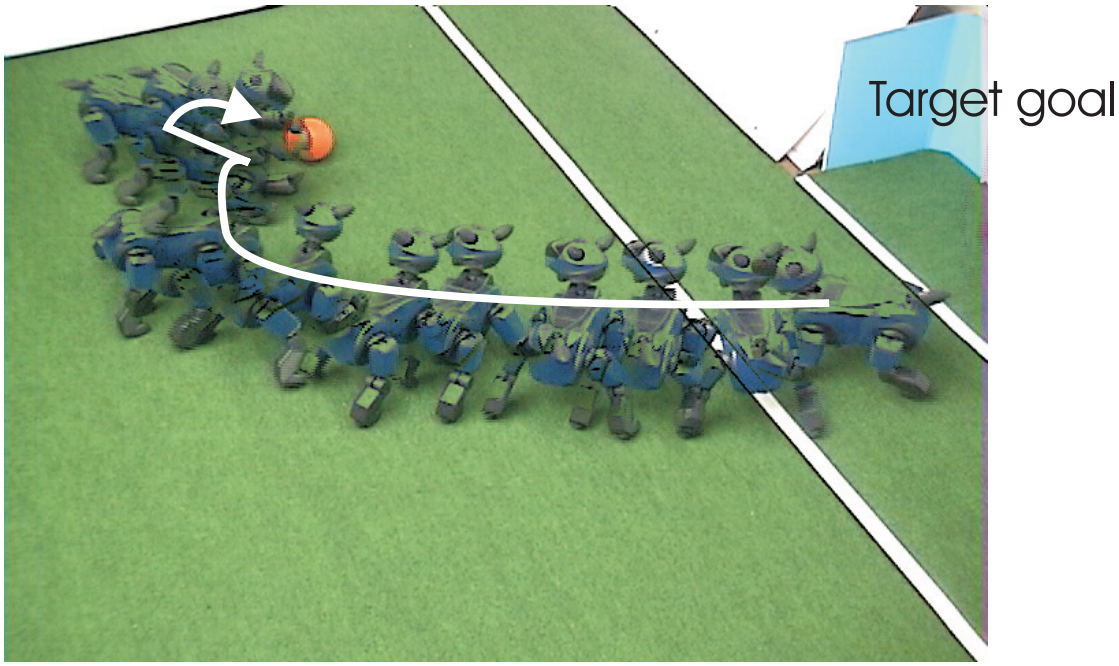

Fig. 7.13　フォワードのボールへの接近行動

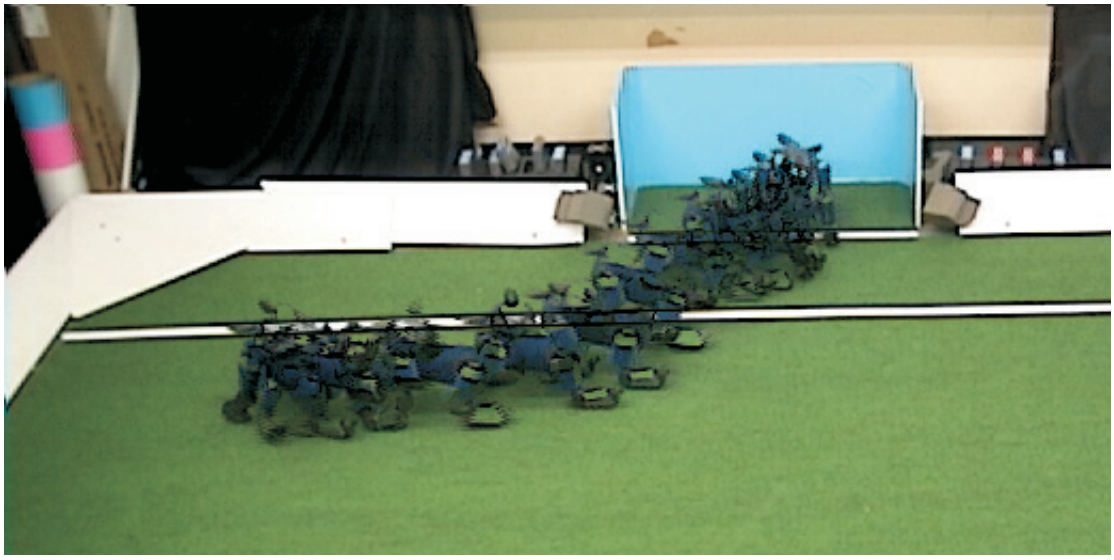

Fig. 7.14　ゴールへの帰還

また，Fig.7.15 では，ゴール前のボールを確保する例である．ロボットはボールを観測し続ける必要があるため，自己位置推定が非常に困難な状況であるが，ロ

ボットの周囲に部分的に観測されたゴールから自己位置を判断して，ボールとゴールポストの間の狭い領域からゴールエリアに入り，ボールに近づいている．

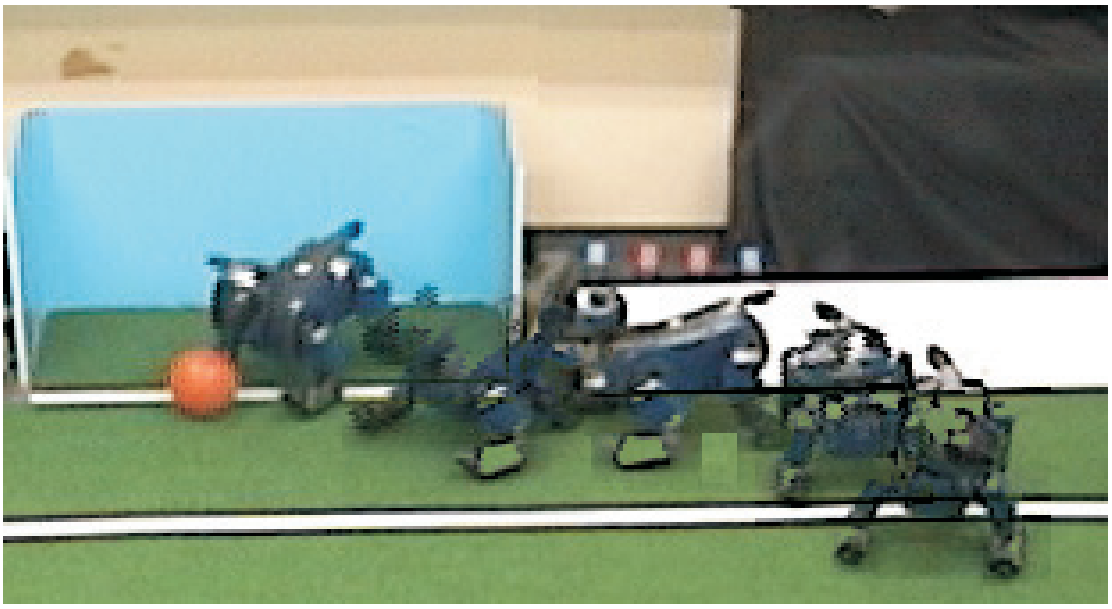

Fig. 7.15　ボール位置を考慮しながらのゴールへの帰還

7.4 おわりに

本章では，実機実験によって，提案した自己位置同定法と，行動決定法の解析，評価を行った．主要な結果をまとめると以下のようになる．

- 自己位置推定について
  - 定点観測によって得られる精度は，233[mm]，6.5[deg]．
  - 歩行時に得られる精度は，333[mm]，14.0[deg]．
  - 照明条件が変化して十分な情報が得られないとき，それがサンプルの分布に反映されて，曖昧に自己位置を表現できる．
  - 情報が十分なときは，曖昧さの表現能力が落ちる．
  - 小さな変位に対して，確率分布を無駄に広くしないで，推定位置が修正される．
  - 強制移動後に，観測ですばやく推定位置が修正される．
- 行動決定法について
  - 提案手法を実装されたキーパーは，得られる情報の質に対して，柔軟にサブタスクを変えて，全体のタスクを達成した．
  - 曖昧さを考慮しない方法を実装されたキーパーは，信頼の低い情報を信じて，無理にタスクを達成しようとした（ゴール脇への衝突等）．

自己位置推定アルゴリズムに関しては，精度の面（233[mm]，6.5[deg]）では，4 足ロボットリーグにおいては，平均的なものであるが，どのような状況においても，得られた情報量に応じた推定を行うという点で優れており，また，現在の 4 足ロボットリーグよりも過酷な照明条件において，有効に機能することを確認した．また，行動決定法に関しては，提案手法を用いることで，自己位置推定アルゴリズムの曖昧な出力を利用することで，得られる情報に応じて行動を変化させるキーパーロボットを実現可能であることを示した．

第8章　結論と今後の展望

8.1　結論

本研究では，4 足ロボットリーグのサッカーフィールドにおいて，自律移動ロボットが，環境の状態（ロボットの自己位置）を推定するアルゴリズムと，その推定結果とオフライン計画の結果から行動決定するアルゴリズムを提案，設計，実装した．

推定アルゴリズムでは，従来は推定に反映することを考慮されていなかった，オクルージョンなどの大きな誤差や，定常的な誤差を発生させる原因に対応するために，ベイズ推定の観測確率モデルを一様分布に限定することを提案し，その方法をモンテカルロ法によって実装する方法: Uniform Monte Carlo Localization（Unform MCL）を示した．また，Unform MCL について，サンプル数を変化させて，精度や自己位置の確率分布の表現力等を調査し，4 足ロボットリーグのフィールドにおいては，1000 個程度のサンプルが必要であると分かった．

Unform MCL をロボットに実装し，実験を行った．実験では，サンプル数 1000 個で定点観測の実験を行い，精度について，位置に対して 233[mm]，方向に対して 6.5[deg] の精度を得られた．また，推定に悪影響を与えるために，照明条件や，ロボットの位置を強制的に変える実験を行ったが，照明条件の変化に対しては，得られる情報なりに自己位置を表現できることを示し，位置変化に対しては，観測によってすばやく，無駄のない推定位置の修正ができることを確認した．この結果から，このアルゴリズムが大きな誤差，定常誤差の要因に対応しながら，効率，精度を損なわないで状態を推定することが可能であることを示した．特に，照明条件の変化の実験については，本研究の課題である，入力される情報の質が変化する「環境の変化」に対応していることが示せた．さらに，計算量の比較から，Uniform MCL が，環境の変化に対応しながら，非常に計算負荷が低い（センサ値入力時 7[ms]，移動時 7.5[ms]）ことを確認できた．

オフライン計画では，動的計画法によって，状態（ロボットの自己位置，ボールの位置）が既知のときにフォワードロボット，キーパーロボットがとるべき行動を求めた．キーパーについては，複数のサブタスクを一つの状態空間で一度に計画を立てた．この計画結果からロボットの行動する軌跡を求めて，ロボットがボールと自己位置に応じてサブタスクを選択することを示した．

オンライン時には，オフライン計画結果とモンテカルロ法のサンプルから，現

在やロボットが行動を起こした後の状態価値の期待値を求め，その期待値が最大になるように行動を選択し続けるアルゴリズムを提案した．そして，それをシミュレーションのエージェントと実機に実装し，シミュレーションと実機によって，提案した行動決定法の評価を行った．シミュレーションでは，提案した行動決定法によって状態価値関数を高めるようにロボットが行動できるかどうかを確認するため，Uniform MCL のサンプルの平均値を推定位置として，その位置における決定論的方策を用いる手法との比較を行った．結果は，提案したものの方が，平均するとロボットが高い状態価値をとり続けるというものであった．

また，提案した行動決定法からは，一つのタスクを遂行するために，どのサブタスクを選択するかを，情報の曖昧さから導くという性質を有することが，理論的に導かれるため，これを確認するために，提案した行動決定をキーパー行動に適用し，実機実験を行った．この実験では，サブタスクのうちの二つについて，どちらを選択しても妥当な状況を与えて，ロボットに行動させて，どちらのサブタスクを選択するかを調べた．このとき，照明を暗くしたり，ゴール板を外したりして，自己位置推定を難しくして，提案手法と，曖昧さを考慮しない手法を比較した．この結果，提案した行動決定法が，得られにくい情報が必要なタスクを避けて，より情報的に容易なサブタスクを選択することで，キーパー行動を遂行しようとすることを確認した．

さらに実機実験で，提案した行動決定法の計算量を調べたところ，平均で 19.8[ms] で，カメラのリフレッシュレートで行動決定するだけの計算速度を記録した．ロボットの CPU の内部周波数は 192MHz であるので，この程度以上の CPU では，十分実時間で処理のできる計算量であることを明らかにした．

以上の結果を Table 1.1 に反映させると，Table 8.1 のようになる．

Table 8.1　各行動決定法の比較

	学習	教示	CN	MC	[深瀬 02]	[横井 01]	提案手法
情報の曖昧さへの対応	○	○	△	○	○	○	○
環境変化に即時対応	×	×	×	×	×	○	○
オフラインメモリ使用量	-	○	△	-	×	○	○
オフライン計算量+人間の作業量	-	△	△	-	△	○	○
オンラインメモリ使用量	○	○	△	×	×	○	○
オンライン計算量	○	○	○	×	○	○	△
複数の行動候補からの選択	○	-	-	○	-	×	○

CN: Coastal navigation, MC: Monte Carlo POMDP.

以上の結果によって，ロボットがタスクを，環境の変化に影響を受けずに遂行するためのアルゴリズムを設計することができた．また，このアルゴリズムによって，タスクの達成のために行うサブタスクを，ロボットが柔軟に変えることを確認した．

8.2 今後の展望

さらにサッカーロボットの性能を向上させたり，より過酷な（実世界に近い）環境で動作させるには，以下のようなことを行う必要がある．

自己位置推定の精度の向上

自己位置推定の精度やロバスト性を向上させるには，画像処理で得られる情報を増やすことが重要である．例えば，フィールド上のラインは，ボールとともに観測できるため，重要な情報源である[*1]．また，観測対象が増加すれば，そのうちのいくつかの情報が誤っていても，他の情報と比較して，よりロバストな自己位置推定が可能となる．特に，人間は，いつも自身の場所を知るために目標としているものが無くなったり，外見が変わっていても，従来の自己位置同定のいくつかのように，位置が全く分からなくなるということは，まず起こらない．このようなときは，他の情報と比較して，その「観測対象」が変化したことを理解する．ロボカップにこの問題を置き換えると，ランドマークの位置が変化していれば，ロボットがその位置変化を認識して，次からその情報を自己位置推定に反映させるようなことである．

ERS-210 のカメラの視野が狭いので，このようなことを行うには，十分な情報が得られないことが，このとき課題となると考えられる．全く未知環境の地図の作成と自己位置同定を同時に行う SLAM（Simultaneous Localization and Mapping）の研究が盛んであるが（[Thrun 00b] など），上記の問題では，それを部分的に応用する手法が考えられそうである．

ボールの位置の確率的表現，多次元空間での推定

現在は，ボールの位置は，ボールが観測される画像から決定論的に求めているが，ボールが観測されていないときでも，ボールの位置が推定できるべきである．これは，自殺点回避や，ボールの探索行動に利用できる．自己位置推定と同様に，モンテカルロ法でこの推定が記述できれば，提案した行動決定法で，ボール位置の曖昧さも考慮して行動決定が行える．ただし，この場合，5 次元空間中に必要なサンプル数の増加に対応する必要があると考えられる．

[*1] これは，[上田 01] で著者が扱ったが，実時間性の問題から現在は使用していない

さらに大きな状態空間への対応（状態価値関数と方策の圧縮）

本研究で提案した手法によって，協調等の高度なタスクを実現するためには，もっと多くの状態変数を扱わなければならない．本論文では，離散状態数をちょうど ERS-210 のメモリの容量にあわせて決定したので，これ以上状態変数が増やすには，方策や状態価値関数を，メモリ容量的に効率よく記述することが必要となる．我々の研究グループでは，方策の圧縮 [Fukase 02, Ueda 03] と，状態価値関数の効率の良い近似方法 [小林 02] に取り組んできたので，これらを利用することを検討している．

観測による状態遷移の考慮

ロボットの行動の性能を向上させるには，観測による状態遷移の考慮が必要である．ただし，従来法のような，観測確率モデルを事前に作成する方法は，序論で述べたように，状態推定アルゴリズムの存在と矛盾するものである．結局，観測確率モデルは，オフライン計画では考慮できず，ロボットが実際に行動することによって獲得せざるを得ないと考えている．

シミュレータを利用した計画

キーパーの計画においては，終端状態が複数存在し，それらの終端状態の価値を筆者が与えたが，この価値の設定を少し誤っただけで，キーパーとして優れた方策が得られない．本研究では，動的計画法が強力な計画手法であるということは示せたが，キーパの計画に成功したのは，筆者が 2 年間キーパーの行動設計を担当していた知識が反映されたからであるかもしれず，誰でも簡単に計画ができるということは示せなかった．この問題に対しては，浅沼氏のシミュレータを用いて計画を立てる方法が有効であると考えている．強化学習の分野では，シミュレータ（計算機）の計算能力と，実機による経験の両方を用いる学習，計画手法が研究されている（たとえば，Dyna-Q など）．より現実に忠実なシミュレータを作成し，シミュレートされたエージェントが高速に学習し，その結果を実機に反映させることが，これから重要になるであろう．また，ロボットの問題に限らず，シミューレーション内で試行錯誤的学習によってあるシステムの最適化を図るというアプローチは，より複雑（多自由度）なシステムに対して有効な手法になっていくと考えている．

シミュレータ自身の学習

上記をさらに発展させた考え方として，シミュレーション自身が実環境を学習するという考え方もできると考えている．現在，ロボットと壁の衝突をモデル化できていないが，これを解析的に行うことは非常に困難であるし，ERS-210 の歩行を改良するたびに解析しなおすのは効率的ではない．これは確率的な考え方を導入して，壁との衝突後のロボットの位置の確率分布を，実環境での少ない試行から予測する方法がないか，現在調査中である．

謝辞

本論文は，東京大学大学院 工学系研究科 精密機械工学専攻 新井・湯浅・太田研究室において執筆したものです．また，実験や作業に関しては，中央大学 理工学部 精密機械工学科 梅田研究室において行いました．執筆において，両研究室の方々に多大なるご指導とご協力を頂きました．ここに，私を支えてくださった方々のお名前を挙げ，あらためて感謝の意を示したいと思います．

新井民夫教授には，非常にお忙しい中，丁寧に指導していただきました．特に，本論文も含めて，論文等の書き方や，国際学会での発表方法などを直接指導していただけたことは，私にとっては非常にありがたいことであり，また実力をつける上で，非常に有意義であったと思います．また，研究以外でも，先生の仕事に対する取り組み方を見ていると，参考になることが多数ありました．

太田順助教授には，研究に関する深い議論をしていただける機会が何度かありました．そのときの議論が，研究の要所で役立つことが何度かあり，大変感謝しております．研究に対する姿勢がどうあるべきかについてアドバイスを頂いたり，私の研究者としての将来を心配していただいたりしました．

故 湯浅秀男助教授は，私が学部 4 年のときの RoboCup グループの指導教官で，今年も一度ミーティングに出席していただきました．また，勉強会においては，直接ご指導をいただけました．何度か数学の質問をしましたが，明快に答えていただき，助かりました．突然のご逝去が悔やまれてなりません．謹んでご冥福をお祈りいたします．

中央大学の梅田和昇教授，大隈久教授には，RoboCup 関係でご協力いただき，大変感謝しております．特に梅田先生には，フィールドの用地の手配や，ミーティングでのアドバイスをしていただけるなど，実験を進める上でお世話になる機会が何度もありました．

助手の前田さんには，今年もここに書き連ねることができないくらい，様々なことでお世話になりました．特に，私は書類が苦手ですので，一つの書類に関して普通の人の何倍も質問をしてご迷惑をおかけしました．また，今年の ICRA で実験に行き詰まり，相談をしていただいたとき，非常に先がすっきりと見えるようになったときは，感動を覚えました．

中央大学修士2年の浅沼 和範君には，本研究で使用されているシミュレータを提供していただきました．本論文の多くの部分が，このシミュレータに依存しており，感謝の言葉もありません．同じ学年で，しかも仕事のできる浅沼君は，僕にとってのよい刺激でした．

中央大学修士1年の神谷 昌吾君には，本研究で使用されている歩行を提供していただきました．この歩行は高速なので，来年一年をRoboCupで戦う上での大きな武器になると思います．

中央大学学部4年の菊地 敏文君には，本研究で用いた色識別テーブルの調整をして頂きました．また，梅田研究室で，よく雑談につきあってくれました．来年は，神谷君や新人，復帰してくる坂本君とともに新潟，Padovaで頂点を目指しましょう．

社会人ドクターの守屋さんには，日立製作所での研究の際にお世話になりました．この研究は，私にとって非常にチャレンジングで面白いものでした．仕事が溜まりっぱなしでお忙しい中，作成したものを何度も見せに行ってすいませんでした．

新井研同期の鈴木君，Trevai君，井澤君，小林君，原君，水田君，修論執筆お疲れ様でした．それぞれとても個性的で，話をしていて面白い人たちばかりでした．これからの活躍を期待しております．

また，博士3年の杉さんを始めとする居室組のみなさんや，元2号館メンバーの方々には，日ごろから雑談や相談に付き合っていただき，大変感謝しております．

全員のお名前を挙げることはいたしませんが，その他の新井研究室，梅田研究室の皆様，秘書の皆様にも，いろいろとお世話になりました．ありがとうございました．

2003年2月　　上田 隆一

参考文献

[Barraquand 95] Jérôme Barraquand and Pierre Ferbach. Motion Planning with Uncertainty: The Information Space Approach. In *Proc. of IEEE International Conference on Robotics and Automation*, 1995.

[Bellman 57] Richard Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[Boyan 95] Justin A. Boyan and Andrew W. Moore. Generalization in Reinforcement Learning: Safely Approximating the Value Function. In *Advances in Neural Information Processing Systems 7*, pp. 369–376, 1995.

[Burgard 96] Wolfram Burgard, Dieter Fox, Daniel Hennig, and Timo Schmidt. Estimating the Absolute Position of a Mobile Robot Using Position Probability Grids. In *In Proc. of the Thirteenth National Conference on Artificial Intelligence*, Vol. 2, pp. 896–901, 1996.

[Burgard 98] Wolfram Burgard, Andreas Derr, Dieter Fox, and Armin B. Cremers. Integrating Global Position Estimation and Position Tracking for Mobile Robots: the Dynamic Markov Lcalization Approach. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1998.

[Buschka 00] P. Buschka, A. Saffiotti, and Z. Wasik. Fuzzy Landmark-Based Localization for a Legged Robot. In *Proc. of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1205–1210, 2000.

[Dellaert 99] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte Carlo Localization for Mobile Robots. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA99)*, pp. 1322–1328, 1999.

[Enderle 01] Stefan Enderle, Marcus Ritter, Dieter Fox, Stefan Sablatnog, Gerhard Kraetzschmar, and Gunther Palm. Vision-based Localization in RoboCup Environments. In *P. Stone, T. Balch, G. Kraetzschmar (Eds.): RoboCup 2000: Robot Soccer World Cup IV*, pp. 291–296, 2001.

[Engelson 92] S. Engelson and D. McDermott. Error correction in mobile robot map learning. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA92)*, pp. 2555–2560, 1992.

[Fox 99] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In *Proc. of AAAI-99*, 1999.

[Fox 00] Dieter Fox, Sebastian Thrun, Wolfram Burgard, and Frank Dellaert. Particle Filters for Mobile Robot Localization. *A. Doucet, N. de Freitas, and N. Gordon, editors, Sequential Monte Carlo Methods in Practice*, pp. 470–498, 2000.

[Fukase 02] Takeshi Fukase, Yuichi Kobayashi, Ryuichi Ueda, Takanobu Kawabe, and Tamio Arai. Real-time Decision Making under Uncertainty of Self-Localization Results. In *Proc. of 2002 International RoboCup Symposium*, pp. 327–379, 2002.

[Gutmann 02] Jens-Steffen Gutmann and Dieter Fox. An Experimental Comparison of Localization Methods Continued. In *In Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 454–459, 2002.

[Hanek 02] Robert Hanek, Thorsten Schmitt, Sebastian Buck, and Michael Beetz. Towards RoboCup without Color Labeling. In *The 2002 International RoboCup Symposiums Pre-Proceedings*, 2002.

[Hengst 01] Bernherd Hengst, Darren Ibbotson, Son Bao Pham, and Claude Sammut. Omnidirectional Locomotion for Quadruped Robots. In *Proc. of RoboCup 2001 International Symposium*, 2001.

[LaValle 00] Steven M. LaValle. Robot Motion Planning: A Game–Theoretic Foundation. *Algorithmica*, Vol. 26, pp. 430–465, 2000.

[Lenser 00] Scott Lenser and Manuela Veloso. Sensor resetting localization for poorly modelled robots. In *Proc. of ICRA-2000*, pp. 1225–1232, 2000.

[Roy 99] Nicholas Roy and Sebastian Thrun. Coastal Navigation with Mobile Robots. In *Advances in Neural Information Processing Systems*, 1999.

[Sutton 98] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[Takahashi 99] Yasutake Takahashi, et al. Continuous Valued Q-learning for Vision-Guided Behavior Acquisition. In *Proc. of the 1999 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 255–260, 1999.

[Thrun 00a] Sebastian Thrun. Monte Carlo POMDPs. *Neural Information Processing Systems*, Vol. 12, pp. 1064–1070, 2000.

[Thrun 00b] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A real-Time Algorithm for Mobile Robot Mapping With Applications to Multi-Robot and 3D Mapping. In *Proc. of the 2000 IEEE International Conference on Robotics and Automation (ICRA2000)*, pp. 321–328, 2000.

[Tsitsiklis 96] John N. Tsitsiklis and Benjamin Van Roy. Feature-Based Methods for Large Scale Dynamic Programming. *Machine Learning*, Vol. 22, pp. 59–94, 1996.

[Tuyls 02] K. Tuyls, et al. Reinforcement Learning in Large State Spaces – Simulated Robotic Soccer as a testbed. In *Proc. of International RoboCup Symposium*, pp. 316–323, 2002.

[Ueda 02] Ryuichi Ueda, Takeshi Fukase, Yuichi Kobayashi, Tamio Arai, Hideo Yuasa, and Jun Ota. Uniform Monte Carlo Localization – Fast and Robust Self-localization Method for Mobile Robots. In *Proc. of ICRA-2002*, pp. 1353–1358, 2002.

[Ueda 03] Ryuichi Ueda, Takeshi Fukase, Yuichi Kobayashi, and Tamio Arai. Vector Quantization for State-Action Map Compression. In *Proc. of ICRA-2003 (to appear)*, 2003.

[Veloso 98] Manuela Veloso, William Uther, Masahiro Fujita, Minoru Asada, and Hiroaki Kitano. Playing Soccer with Legged Robots. In *Proc. of IEEE/RSJ IROS-98*, 1998.

[浅田 97] 浅田稔, 國吉康夫, 野田五十樹, 北野宏明. 研究活動とロボットコンテスト (RoboCup). 日本ロボット学会誌, Vol. 15, No. 1, pp. 1001–1004, 1997.

[浅田 00] 浅田稔, 北野宏明. ロボカップ戦略：研究プロジェクトとしての意義と価値. 日本ロボット学会誌, Vol. 18, No. 8, pp. 1081–1084, 2000.

[浅沼 03] 浅沼和範. カメラ特性を考慮した自律移動ロボットのための環境・計測シミュレータの開発. 中央大学大学院理工学研究科精密工学専攻平成 14 年度修士論文, 2003.

[上田 01] 上田隆一. 四脚ロボットによるサッカー行動生成のための局所環境認識. 平成 12 年度東京大学工学部精密機械工学科卒業論文, 2001.

[大橋 02] 大橋健. ４足ロボットリーグのとりくみ. 日本ロボット学会誌, Vol. 20, No. 1, pp. 45–46, 2002.

[OPE 02] OPEN-R プログラミング SIG 著, ソニー株式会社エンタテイメントロボットカンパニー監修. C++で AIBO を自在に動かす OPEN-R プログラミング入門. インプレス, 2002.

[加藤 87] 加藤寛一郎. 最適制御入門 レギュレータとカルマンフィルタ. 東京大学出版会, 1987.

[北川 67] 北川敏男, 喜安喜市. マルコフ過程. 共立出版, 1967.

[小林 02] 小林祐一. 強化学習のための自律分散型関数近似法. 平成 13 年度博士論文, 東京大学, 2002.

[近藤 84] 近藤次郎. 最適化法. コロナ社, 1984.

[高橋 99] 高橋泰岳, 浅田稔. 実ロボットによる行動学習のための状態空間の漸次的構成. 日本ロボット学会誌, Vol. 17, No. 1, pp. 118–124, 1999.

[野田 92] 野田一雄, 宮岡悦良. 数理統計学の基礎. 共立出版, 1992.

[深瀬 02] 深瀬武. 四足ロボットのための不確かな状況認識下での実時間行動決定法. 平成 13 年度修士論文, 東京大学, 2002.

[Wil 93] William H.Press, Saul A. Teukolsky, William T. Vetterling, Brian P.Flannery 著, 丹慶勝市, 奥村晴彦, 佐藤俊郎, 小林誠 訳. Numerical Recipes in C [日本語版] C 言語による数値計算のレシピ. 技術評論社, 1993.

[松原 02] 松原仁, 浅田稔, 北野宏明. ロボカップの歴史と 2002 年への展望. 日本ロボット学会誌, Vol. 20, No. 1, pp. 2–6, 2002.

[光永 01] 光永法明, 浅田稔. 移動体の意思決定のための情報量基準に基づく観測対象戦略. 日本ロボット学会誌, Vol. 19, No. 6, pp. 793–800, 2001.

[横井 01] 横井真浩. 四脚ロボットにおける観測コストを考慮したナビゲーション. 平成 12 年度修士論文, 東京大学, 2001.

研究業績

学位論文

- 上田 隆一: “四脚ロボットによるサッカー行動生成のための局所環境認識”, 平成 12 年度東京大学精密機械工学科卒業論文, 2001.

査読つき講演論文

- Takeshi Fukase, Masahiro Yokoi, Yuichi Kobayashi, Ryuichi Ueda, Hideo Yuasa and Tamio Arai: “Quadruped Robot Navigation Considering the Observational Cost,” Andreas Birk, Silvia Coradeschi and Satoshi Tadokoro (Eds.), RoboCup 2001: Robot Soccer World Cup V, pp. 350-355, Springer, 2002.

- Ryuichi Ueda, Takeshi Fukase, Yuichi Kobayashi, Tamio Arai, Hideo Yuasa, and Jun Ota: “Uniform Monte Carlo Localization - Fast and Robust Self-localization Method for Mobile Robots,” Proc. of International Conference on Robotics and Automation (ICRA), pp. 1353-1358, 2002.

- Takeshi Fukase, Yuichi Kobayashi, Ryuichi Ueda, Takanobu Kawabe and Tamio Arai: “Real-time Decision Making under Uncertainty of Self-Localization Results,” The 2002 International RoboCup Symposium Pre-Proceedings, pp. 372-379, 2002.

- Ryuichi Ueda, Takeshi FUKASE, Yuichi KOBAYASHI and Tamio ARAI: “Vector Quantization for State-Action Map Compression,” Proc. of International Conference on Robotics and Automation (ICRA), Taipei, Taiwan, 2003. (to appear)

- 上田 隆一，深瀬武，小林祐一，新井民夫, “ベクトル量子化による状態・行動地図の圧縮”, 第 8 回ロボティクス・シンポジア，静岡，2003. (to appear)

口頭発表

- 上田 隆一，小林 祐一，横井 真浩，深瀬 武，湯浅 秀男，新井 民夫: “四脚ロボットにおける Monte Carlo Localization に基づく環境認識”, 日本機械学会ロボティクス・メカトロニクス講演会（ROBOMEC01），香川，2001.
- 上田 隆一，小林 祐一，深瀬 武，新井 民夫，湯浅 秀男，太田 順: “一様分布に基づく高速モンテカルロ自己位置同定”, 日本ロボット学会学術講演会, pp. 1007-1008，2001.
- 上田 隆一，トリワイ チョンチャナ，守屋 俊夫，新井 民夫: “移動ロボットによる広域シームレス映像の取得–直線マーカを利用したモザイキングとスーパーレゾリューション”, 精密工学会春季大会学術講演会, 2003. (to appear)
- トリワイ チョンチャナ，上田 隆一，守屋 俊夫，新井 民夫: “移動ロボットによる広域シームレス映像の取得–移動ロボットの作業領域における自己位置同定”, 精密工学会春季大会学術講演会, 2003. (to appear)

付録 A　サッカープログラムの実装

A.1　はじめに

ここでは，研究に用いたERS-210のサッカープログラム全体について，簡単に解説する．また，コードを簡潔に書いたり，計算処理を高速化するために行った工夫について説明する．ただし，工夫については，ERS-210や，MIPSの仕様に左右されるものが存在する．

参考までに，本論文の手法のいくつかが利用されたRoboCup用のプログラムが，http://www.arai.pe.u-tokyo.ac.jp/RoboCup/index.html に公開されている．そして，ロボットの実際の行動等を http://prince.pe.u-tokyo.ac.jp/~ ueda/に mpeg形式で公開している．

A.2　全体の構成

クラスは，以下のように分類されている．主要なクラスの構成を Fig.A.1 に示す．

- 最重要クラス
  - SoccerLion: センサ値を得て，動作の指令を出す．
  - State: ロボットのセンサ値と現在の動作，認識した状態を管理する．
- 情報処理部
  - CDT: カメラ画像から，色抽出画像を作成する．
  - Find: フィールド上のある特定の物体を，画像から抽出する．
  - SampleOperator: Uniform MCL のサンプルを管理，操作する．
- 行動決定部
  - Behavior: 頭部と歩行の動きを決定して，コントローラー（後述）に指令を出す．
  - Strategy: Behavior を切り替える．
- コントローラー
  - HeadController: Behavior が与える頭部へ与える指令を，現在の頭部の動きから，許可するかどうか判断して SoccerLion に伝える．
  - WalkController: 利用可能な歩行要素を記録している．Behavior が与える歩行の指令に不正がないかチェックして SoccerLion に伝える．

最重要なものは，図中の SoccerLion と state である．この図中の他のクラスは，全て SoccerLion のインスタンスとなっている．SoccerLion に直属する関数，変数は，ロボットのセンサ値を読み取り，ロボットのアクチュエータに指令を出すために用いられる．

State は，ある時点でのロボットのセンサ値や，センサ値から得られた情報が全て記録される巨大な静的クラスである．他のクラスからは，State の中のデータを自由に得ることができるようになっている．一方，State の中のある特定のデータを更新する場合には，更新できるクラスが厳密に決められており，State 以外のクラス（インスタンス）が直接データのやり取りを行うことは，原則的に禁止してい

る[*1]．そうすることで，プログラムが混乱することを避けることができる．State の値を更新できるのは，原則として，SoccerLion 本体と，自己位置推定に用いられる SampleOperator から派生したクラス，フィールド上の各物体を画像から発見するための，Find から派生したクラス，色抽出画像を作成するクラス CDT である．

行動決定部分は，State の値を読んで，頭部の動作や歩行要素を決定する．この部分は，Strategy と Behavior に分かれており，現在は Strategy が Behavior を，現在の状態から切り替える役割をしている．頭部の動作や歩行要素を直接決定するのは，Behavior で，ここに本研究で設計した DP による行動決定法や，他の行動決定法が実装されている．図中の Behavior は，本研究に関連するもののみで，実際には約 20 種類程度存在する．

[*1]ただし，本研究では，Behavior が直接サンプルを用いて行動決定するため，SampleOperator で管理されているサンプルを静的変数にして，直接に値を参照したため，上記の原則から外れている．

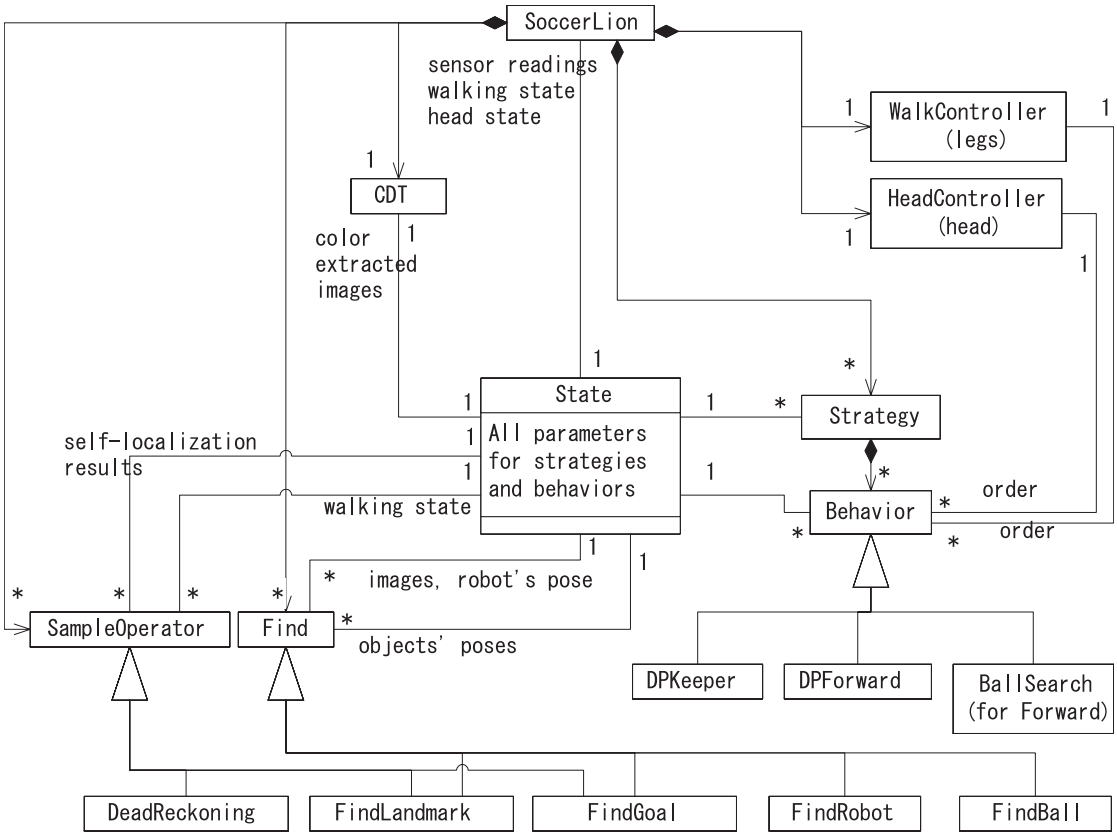
SoccerLion
sensor readings
walking state
head state
WalkController
(legs)
CDT
color
extracted
images
HeadController
(head)
State
All parameters
for strategies
and behaviors
Strategy
self-localization
results
walking state
Behavior
order
order
images, robot's pose
SampleOperator
Find
objects' poses
DPKeeper
DPForward
BallSearch
(for Forward)
DeadReckoning
FindLandmark
FindGoal
FindRobot
FindBall

Fig. A.1　クラス図

A.3 Uniform MCL 関係のクラス

自己位置推定に関係するクラスは，サンプルを管理する静的クラスを一つ用意して，デッドレコニングのクラスと，ランドマークとゴールの位置計測に関するクラスがそれを継承する形を採用している．Fig.A.2 に，Fig.A.1 から，これらの関係を抜粋する．SampleOperator がサンプルを管理するクラスで，ここにはサンプルを操作するための仮想関数 UpdateSample() が定義されている．SampleOperator からは，図のように三つのクラスが派生している．FindLandmark と FindGoal は，色抽出画像からランドマーク，ゴールを見つけ出し，UpdateSample() でサンプル消去が実装される．また，DeadReckoning の UpdateSample() では，サンプルの移動が実装される．

図中の x, y, θ,isAlive が，サンプルについての変数で，それぞれサンプル数の上限の個数の要素を持つ配列になっている．(x, y, θ) がサンプルの位置，向きである．また，isAlive は，そのサンプルが現在有効か，無効かを示しており，サンプルを消去するときは，そのサンプルの isAlive を false にして，サンプル数 NumberOfSamples を一つ減少させる．本文の数式ではサンプルは ξ_i $(i = 1, 2, \ldots, N_{\mathrm{sample}})$ と表現されているが，サンプルを消去するたびにインデックス i を付け直すことは無駄であるので，実装では，このような方法でサンプルを消去する．

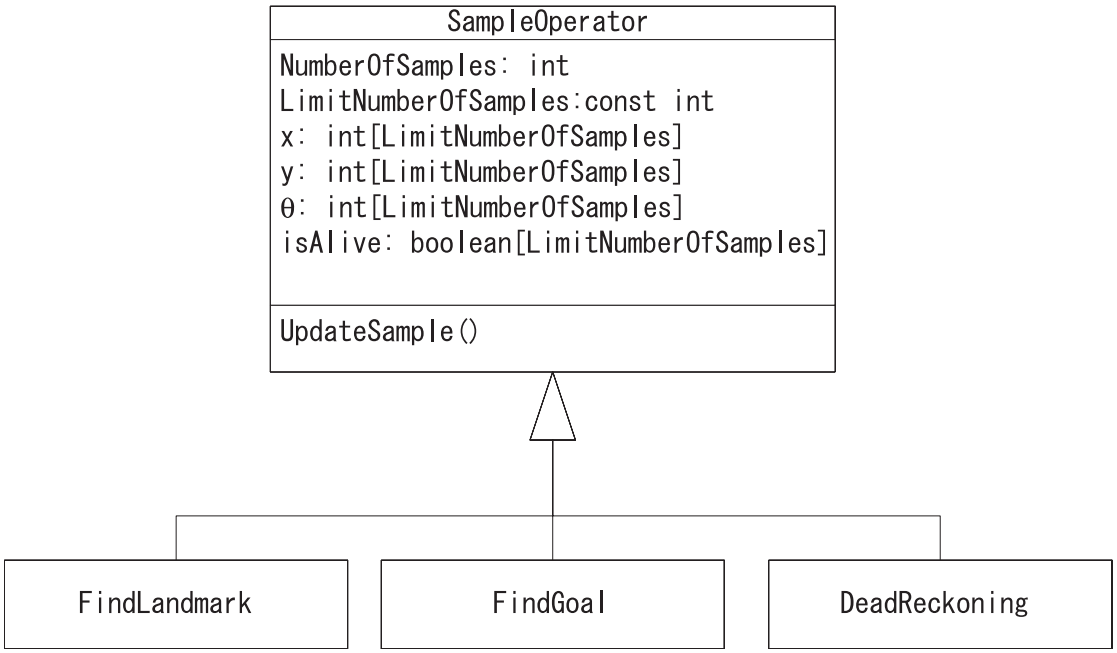

Fig. A.2　自己位置推定に関するクラス

A.4 Uniform MCLの高速化の工夫

A.4.1 配列への書き出し

乱数，三角関数は，コンストラクタで配列に書き出し，プログラム中ではその配列を利用すると，標準C++ライブラリのrand(),sin(),cos()の関数を使用することと比較して，3-10倍の速度で処理ができる．

公開している試合用のコードでは，乱数については，4096（0x1000）個を配列にしている．ただ，この数は，1000個のサンプルに対しては少ない[*2]が，実用上は問題がない．

A.4.2 角度の表現

自己位置推定では，ロボットの方向に関する計算において，角度の計算が頻出する．これらの計算では，-180[deg]と180[deg]（あるいは0[deg]と360[deg]）を，プログラム上でつなぎ合わせる必要がある．普通にこれらの角度の処理をコード化すると，if文や，正規化の処理が必要となり，高速化を阻害する要因となる．また，角度の処理は，バグの原因となりやすい．

そこで，サンプルのθ値は，符号なし整数型を利用して，次のような工夫を行って表現している．まず，360[deg]を512等分する．すると，9bitで角度を表現することができる．実際には，2byteの符号なし整数型（ERS-210の場合はunsigned short）の下9桁を用いて角度を表現する．1byteにしないのは，256等分であると，細かい角度の表現に不十分だからである．

このようにすると，0[deg]から角度を引いても，下9桁では正確に角度が計算できている．正規化した角度が得たいならば，マスク0x1FFとのandをとる単純な計算でよい．また，A.4.1節での三角関数の配列も，それにあわせて512個の要素を持つように作成すると，sin(),cos()の呼び出しで，ラジアンに直す必要がないので，よりいっそう効果的である[*3]．

[*2]標準ライブラリが用意する乱数発生用の関数には十分に注意すべきであることが，[Wil 93]に詳細に記述されているので，参考にされたい．

[*3]360[deg]で表現するなら，最初からラジアンにするべきである．しかし，実感がつかみにくくプログラムしにくいという欠点もある．

A.5　その他の要点

A.5.1　角度に関する平均値と分散の算出方法

θ 軸における Uniform MCL のサンプルの分布について，平均値と分散を求める方法を示す．

(1) サンプルの存在しない区間で，最長の部分を求める．

(2) この区間の補集合を，サンプルの存在する区間 $[\theta_{\mathrm{right}}, \theta_{\mathrm{left}}]$ とする．

(3) θ_{right} を仮に 0[deg] として，サンプル位置の平均値と，平均値まわりの分散を求める．

(4) 求めた平均値に θ_{right} を足して，θ 軸におけるサンプル分布の平均値とする．分散は，そのままでよい．